QS STARS RATED FOR EXCELLENCE 2012

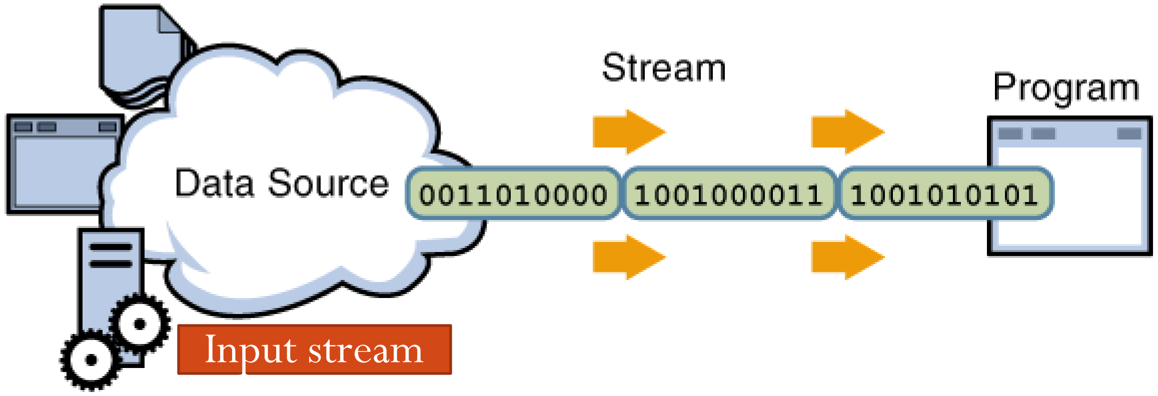
Fpt University

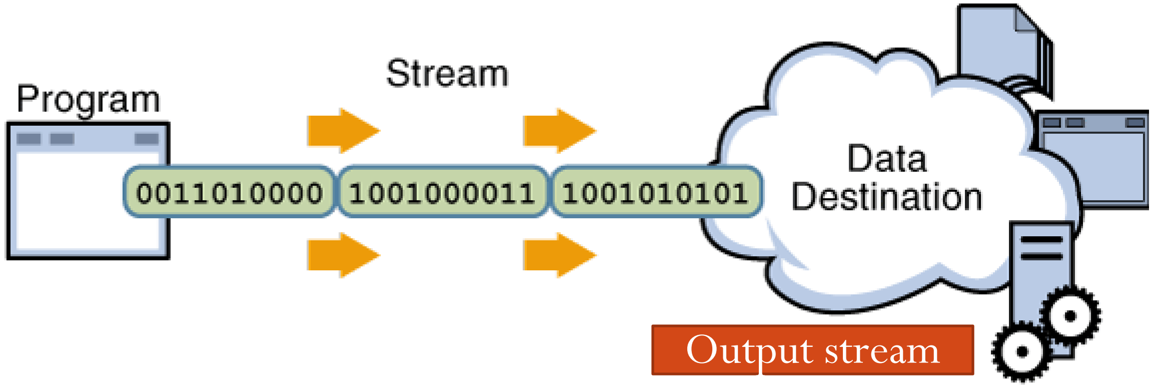
TRƯỜNG ĐẠI HỌC FPT

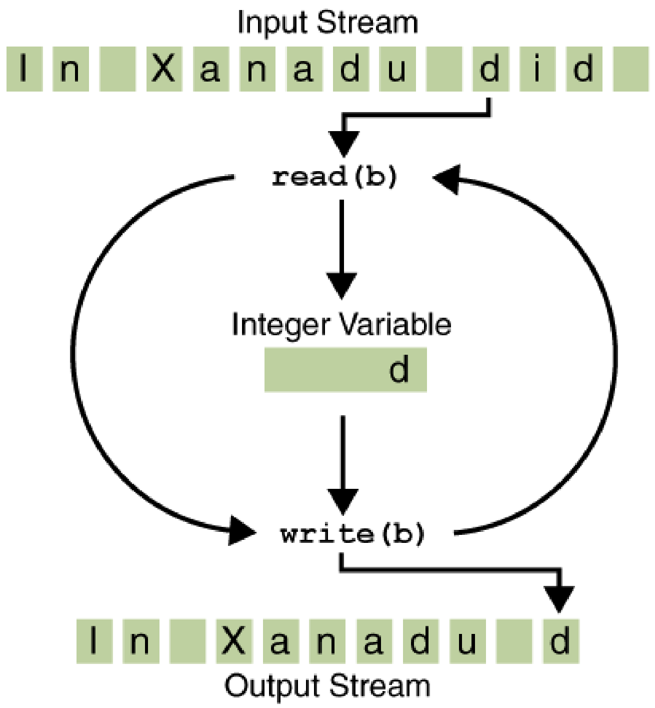
File I/O

What are streams?

- A stream is an object managing a data source in which operations such as read data in the stream to a variable, write values of a variable to the stream associated with type conversions are performed automatically. These operations treat data as a chain of units (byte/character/data object) and data are processed in unit-by-unit manner.







Why should you study this chapter?

- Files can not be missing in large applications.
- Do you want to access a file in Java?
- How can we read/write data from/to a file?

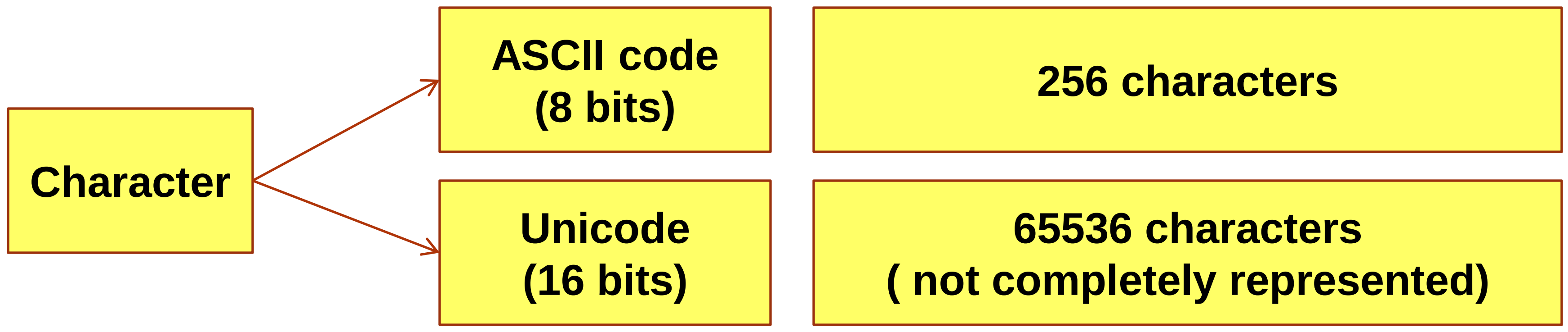
Objectives

- Distinguishing Text, UTF, and Unicode
- How to access directories and files?
- How to access text files.
- How to access binary files?
- How to read/write objects from/to files

Contents

- Text, UTF, and Unicode
- Introduction to the java.io package
- Accessing directories and files
- Accessing binary files
- Accessing text files.
- Read/write objects from/to files?

1- Text, UTF, and Unicode



Unicode character: a character is coded using 16/32 bits

UTF: **U**niversal Character Set – UCS- **T**ransformation **F**ormat

UTF: *Unicode transformation format , a* Standard for compressing strings of Unicode text .

UTF-8: A standard for compressing Unicode text to 8-bit code units.

Refer to: http://www.unicode.org/versions/Unicode7.0.0/

Java :

- Uses UTF to read/write Unicode
- Helps converting Unicode to external 8-bit encodings and vice versa.

2- Introduction to the java.io Package

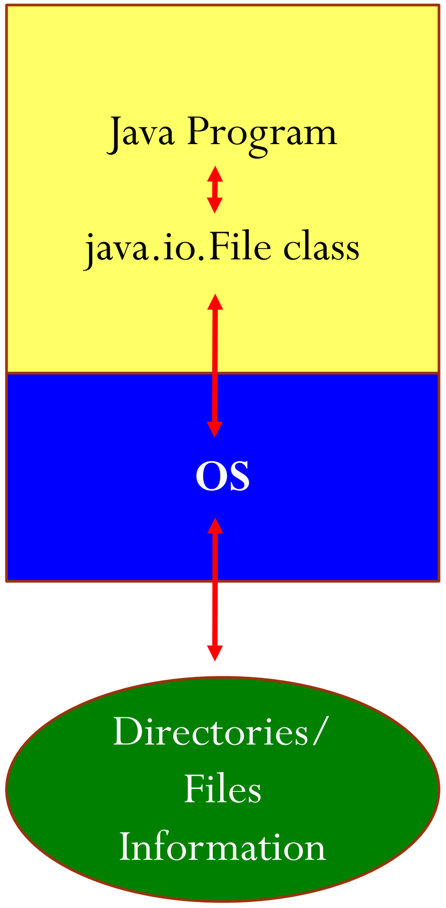
- Java treats all data sources (file, directory, IO devices,…) as streams
- The java.io package contains Java APIs for accessing to/from a stream.
- A stream can be a binary stream.
  - Binary low-level stream: data unit is a physical byte.
  - Binary high-level stream: data unit is primitive data type value or a string.
  - Object stream: data unit is an object.
- A stream can be a character stream in which a data unit is an Unicode character.

TRƯỜNG ĐẠI HỌC FPT

3- Accessing directories and files

The java.io.File Class

Class represents a file or a directory managed by operating system.



Constructor Summary

File(File parent, String child)
Creates a new File instance from a parent abstract pathname and a child pathname string.

File(String pathname)
Creates a new File instance by converting the given pathname string into an abstract pathname.

File(String parent, String child)
Creates a new File instance from a parent pathname string and a child pathname string.

File(URI uri)
Creates a new File instance by converting the given file: URI into an abstract pathname.

Accessing directories and files…

The java.io.File Class…

Common Methods:

boolean canExecute(), canRead(), canWrite()
boolean exists(), isDirectory(), isFile()
String getAbsolutePath(), getCanonicalPath(),
getName(), getParent()
String[] list()
boolean delete(), createNewFile(), mkDir(),
rename(File newName)
long length()

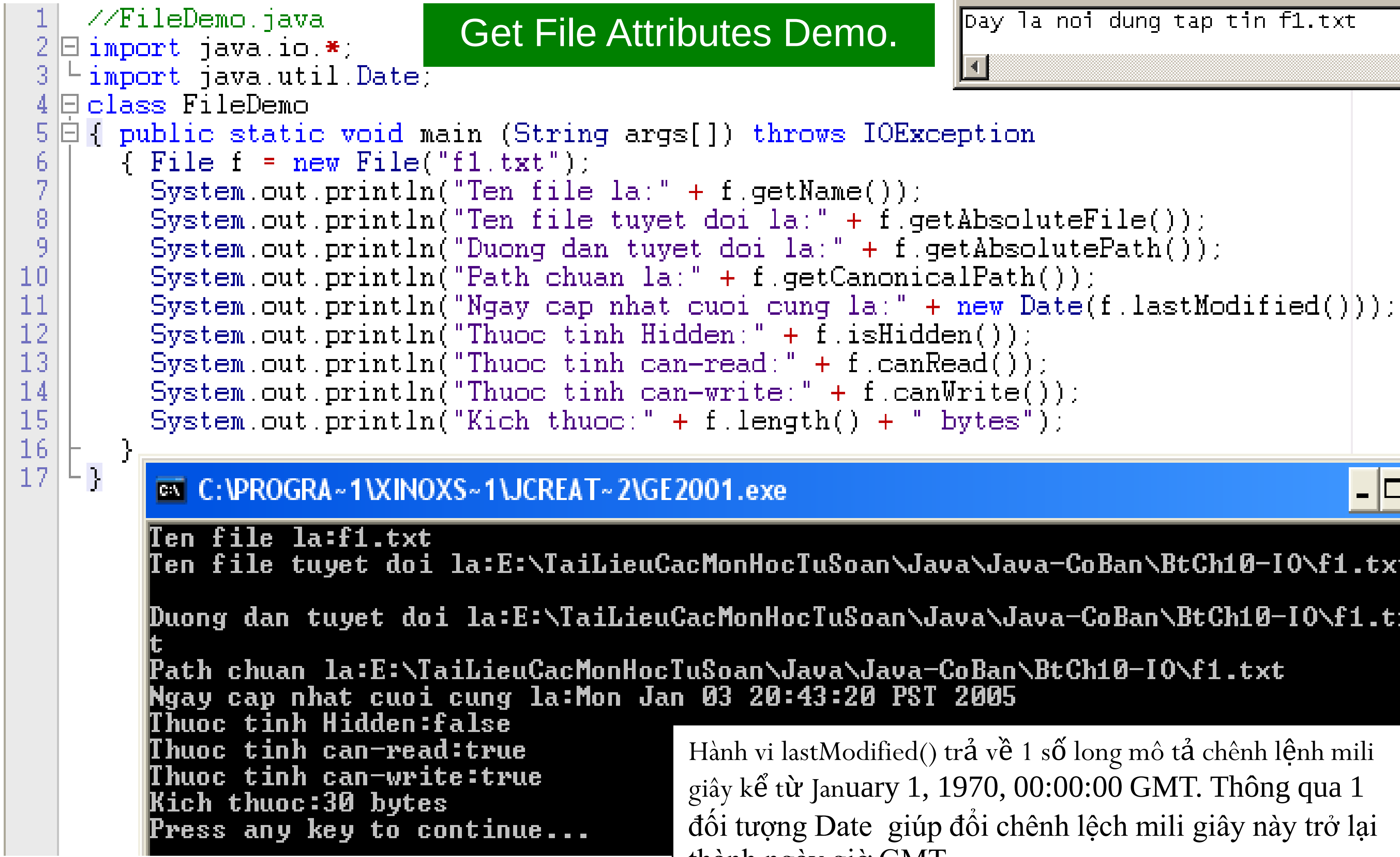
This class helps accessing file/directory information only. It does not have any method to access data in a file.

Method Invoked	Returns on Microsoft Windows	Returns on Solaris
getAbsolutePath()	c:\java\examples\examples\xanadu.txt	/home/cafe/java/examples/xanadu.txt
getCanonicalPath()	c:\java\examples\xanadu.txt	/home/cafe/java/examples/xanadu.txt

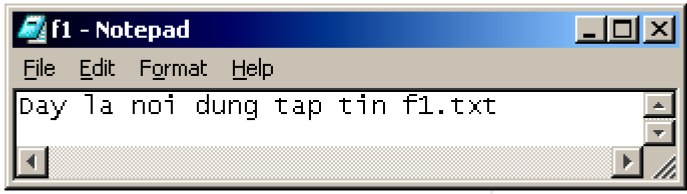
Accessing directories and files…

The java.io.File Class…

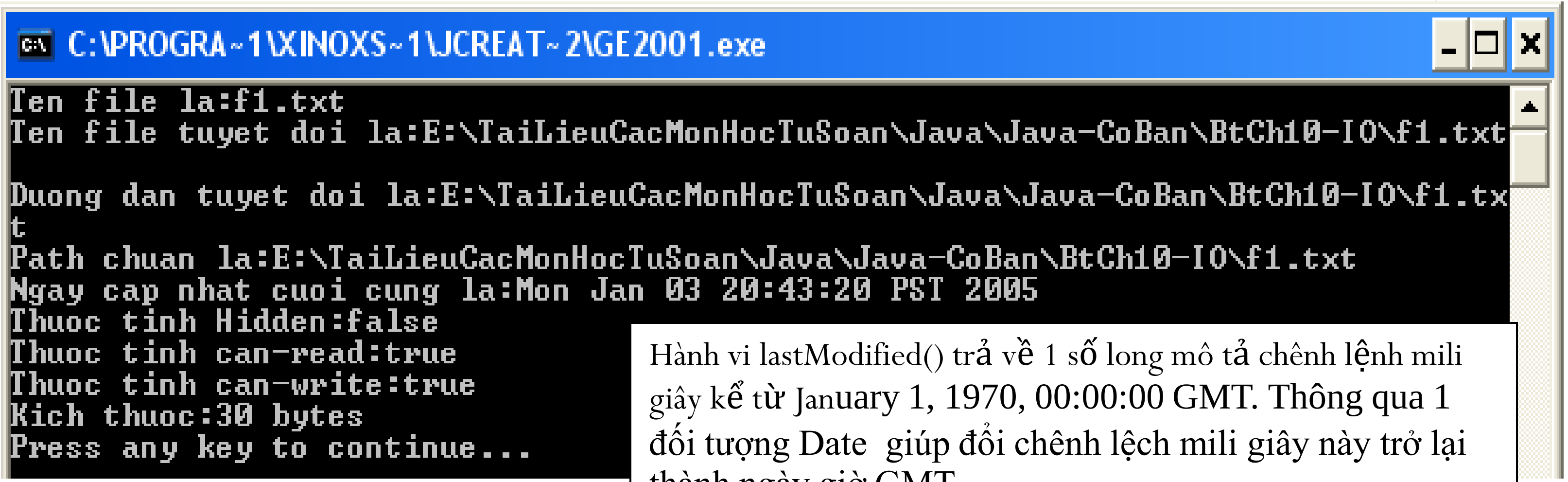
Get File Attributes Demo.



```java
//FileDemo.java
import java.io.*;
import java.util.Date;
class FileDemo
{ public static void main (String args[]) throws IOException
  { File f = new File("f1.txt");
    System.out.println("Ten file la:" + f.getName());
    System.out.println("Ten file tuyet doi la:" + f.getAbsoluteFile());
    System.out.println("Duong dan tuyet doi la:" + f.getAbsolutePath());
    System.out.println("Path chuan la:" + f.getCanonicalPath());
    System.out.println("Ngay cap nhat cuoi cung la:" + new Date(f.lastModified()));
    System.out.println("Thuoc tinh Hidden:" + f.isHidden());
    System.out.println("Thuoc tinh can-read:" + f.canRead());
    System.out.println("Thuoc tinh can-write:" + f.canWrite());
    System.out.println("Kich thuoc:" + f.length() + " bytes");
  }
}
```



```
Day la noi dung tap tin f1.txt
```



```
Ten file la:f1.txt
Ten file tuyet doi la:E:\TaiLieuCacMonHocTuSoan\Java\Java-CoBan\BtCh10-IO\f1.txt

Duong dan tuyet doi la:E:\TaiLieuCacMonHocTuSoan\Java\Java-CoBan\BtCh10-IO\f1.tx
t
Path chuan la:E:\TaiLieuCacMonHocTuSoan\Java\Java-CoBan\BtCh10-IO\f1.txt
Ngay cap nhat cuoi cung la:Mon Jan 03 20:43:20 PST 2005
Thuoc tinh Hidden:false
Thuoc tinh can-read:true
Thuoc tinh can-write:true
Kich thuoc:30 bytes
Press any key to continue...
```

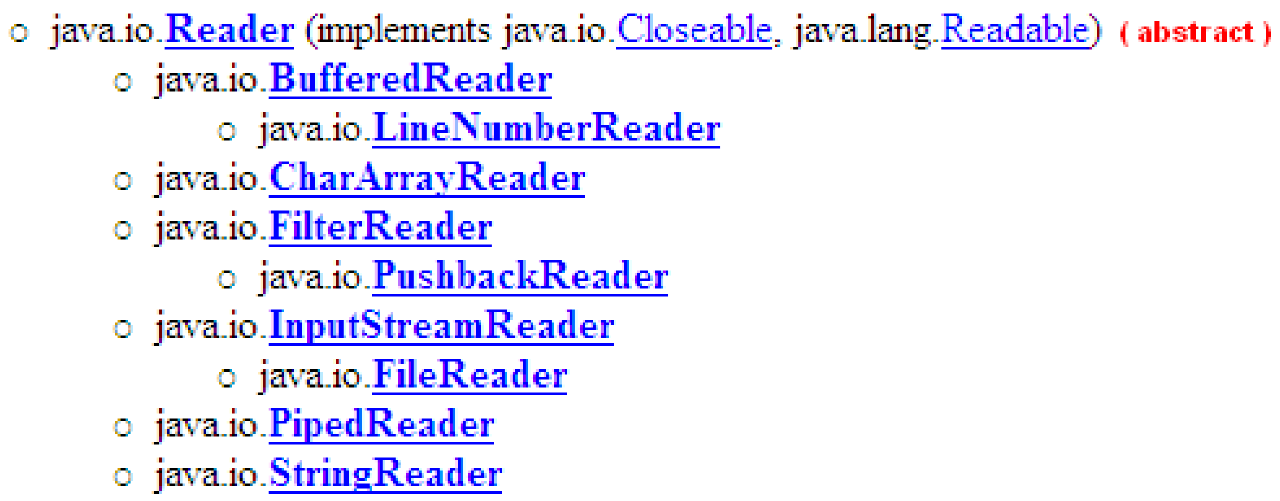
Hành vi lastModified() trả về 1 số long mô tả chênh lệnh mili giây kể từ January 1, 1970, 00:00:00 GMT. Thông qua 1 đối tượng Date giúp đổi chênh lệch mili giây này trở lại thành ngày giờ GMT

4- Access Text Files

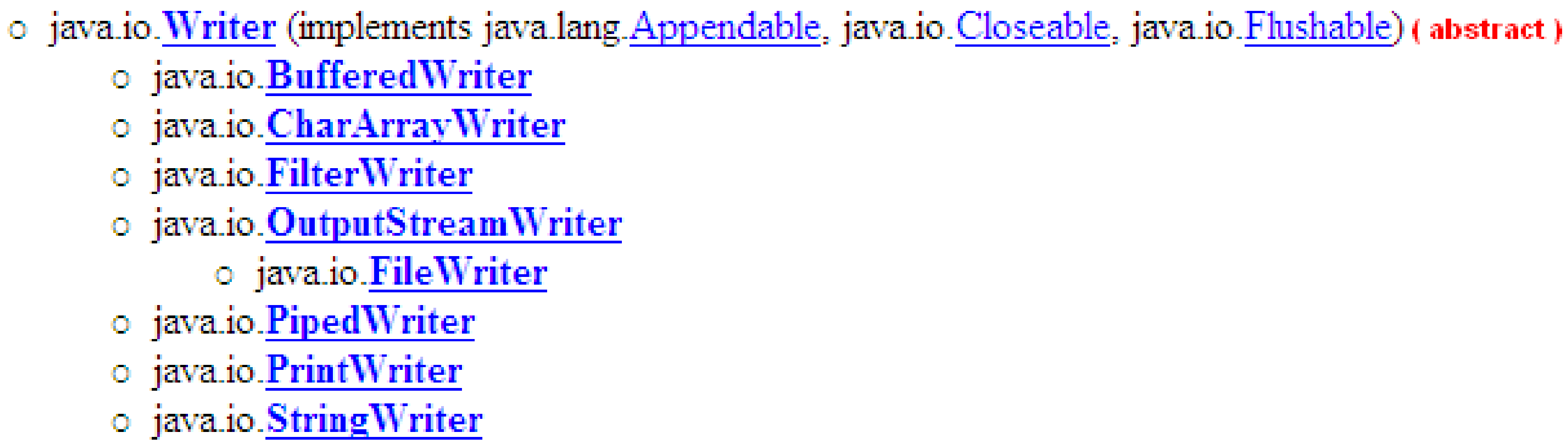
- Character Streams:
  - Two ultimate abstract classes of character streams are Reader and Writer.
  - Reader: input character stream will read data from data source (device) to variables (UTF characters).
  - Writer: stream will write UTF characters to data source (device).

TRƯỜNG ĐẠI HỌC FPT

Access Text Files … Character Streams



- java.io.Reader (implements java.io.Closeable, java.lang.Readable) (abstract)
  - java.io.BufferedReader
    - java.io.LineNumberReader
  - java.io.CharArrayReader
  - java.io.FilterReader
    - java.io.PushbackReader
  - java.io.InputStreamReader
    - java.io.FileReader
  - java.io.PipedReader
  - java.io.StringReader

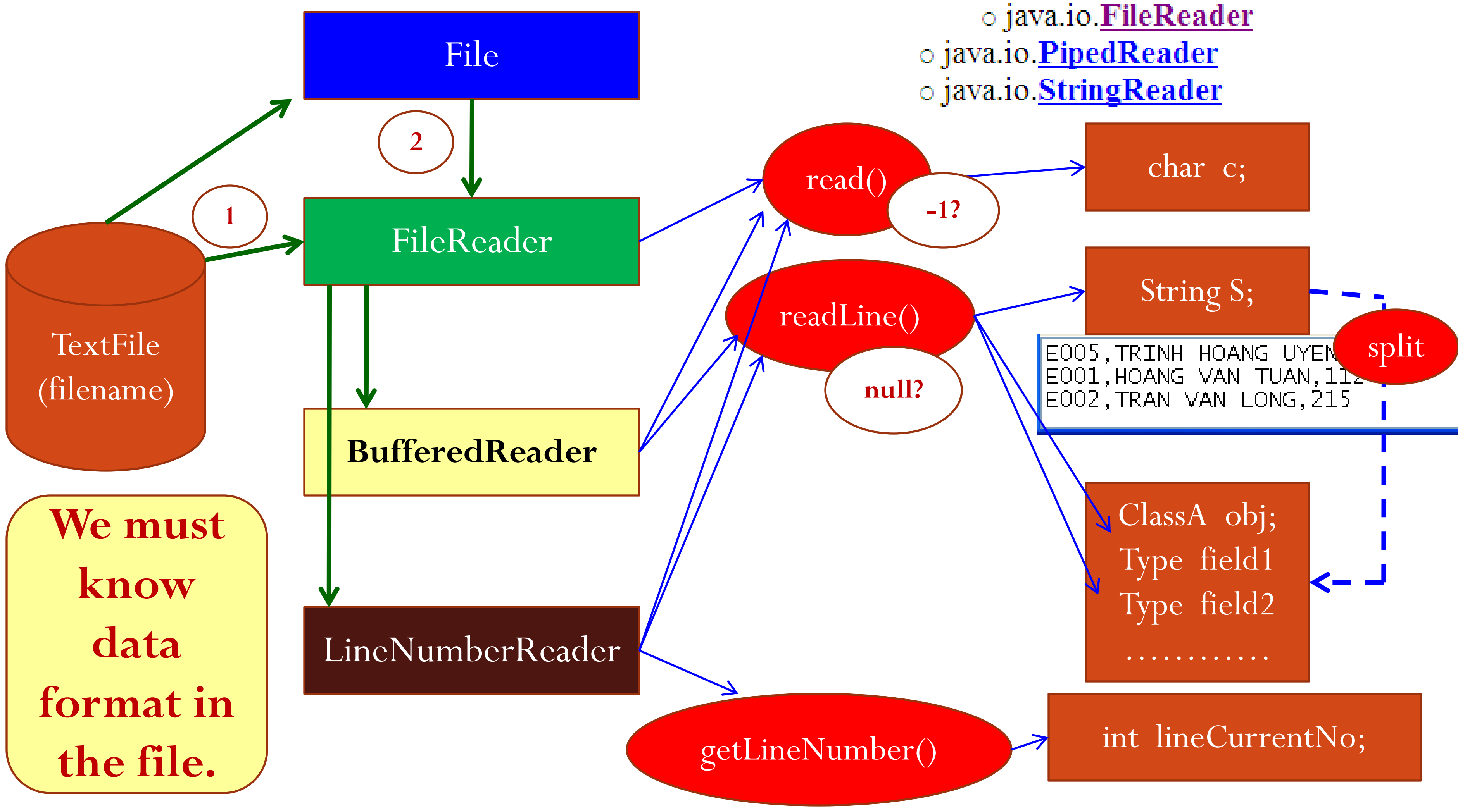


- java.io.Writer (implements java.lang.Appendable, java.io.Closeable, java.io.Flushable) (abstract)
  - java.io.BufferedWriter
  - java.io.CharArrayWriter
  - java.io.FilterWriter
  - java.io.OutputStreamWriter
    - java.io.FileWriter
  - java.io.PipedWriter
  - java.io.PrintWriter
  - java.io.StringWriter

TRƯỜNG ĐẠI HỌC FPT

Access Text Files …
Reading Data

- java.io.Reader
  - java.io.BufferedReader
    - java.io.LineNumberReader
  - java.io.CharArrayReader
  - java.io.FilterReader
    - java.io.PushbackReader
  - java.io.InputStreamReader
    - java.io.FileReader
  - java.io.PipedReader
  - java.io.StringReader

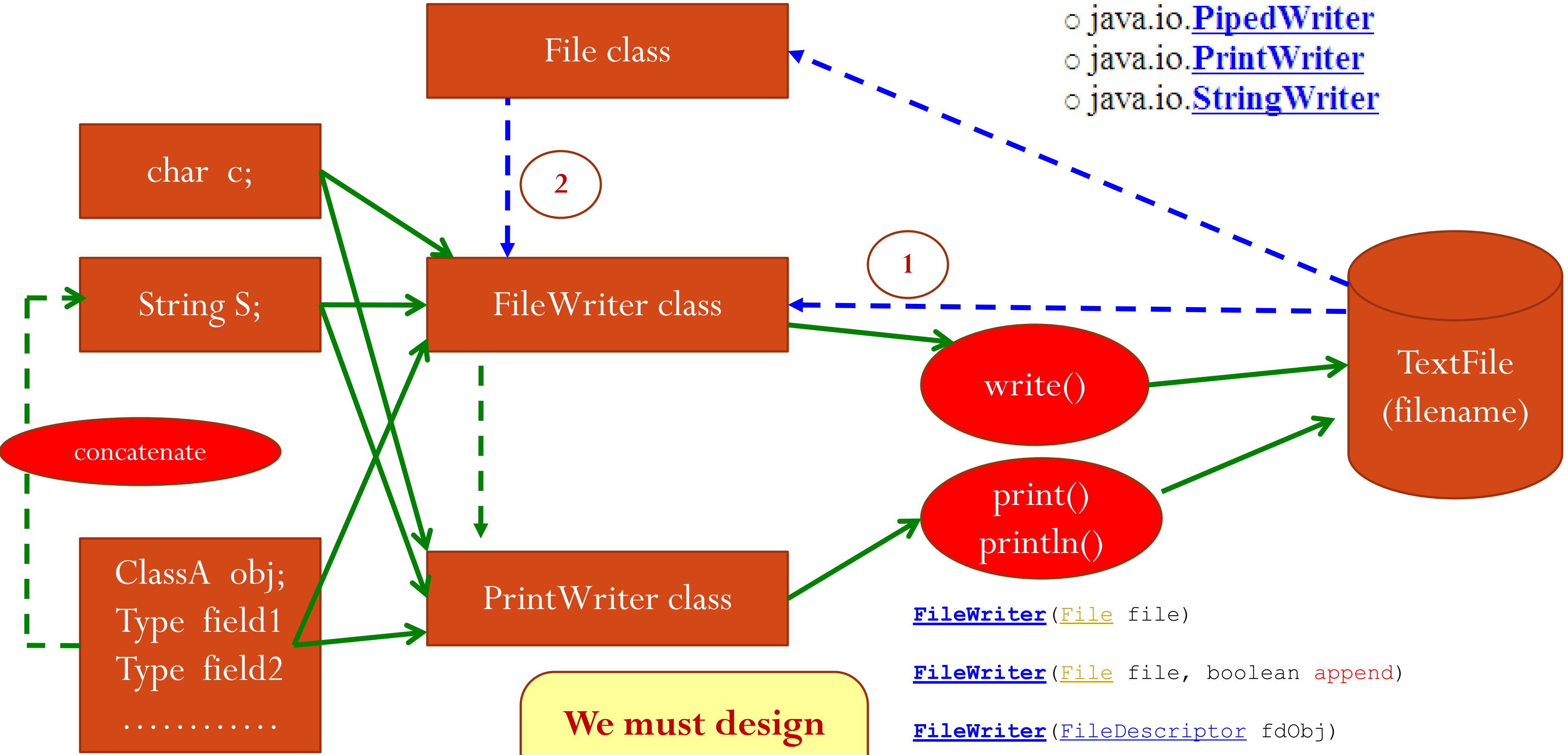


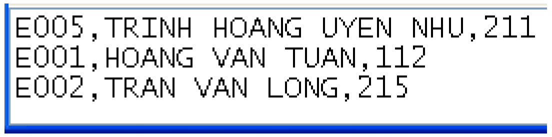
```
E005,TRINH HOANG UYEN
E001,HOANG VAN TUAN,112
E002,TRAN VAN LONG,215
```

We must know data format in the file.

Access Text Files … Writing Data

- java.io.Writer
  - java.io.BufferedWriter
  - java.io.CharArrayWriter
  - java.io.FilterWriter
  - java.io.OutputStreamWriter
    - java.io.FileWriter
  - java.io.PipedWriter
  - java.io.PrintWriter
  - java.io.StringWriter





```
E005,TRINH HOANG UYEN NHU,211
E001,HOANG VAN TUAN,112
E002,TRAN VAN LONG,215
```

We must design the data format in the file.

```
FileWriter(File file)
FileWriter(File file, boolean append)
FileWriter(FileDescriptor fdObj)
FileWriter(String name)
FileWriter(String name, boolean append)
```

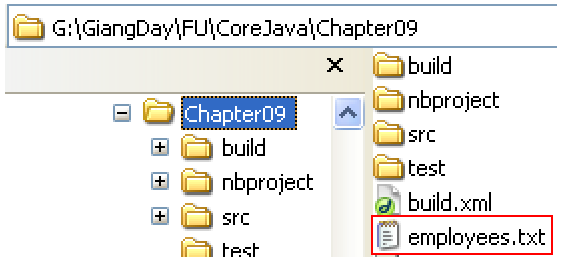
Access Text Files … Case study 1

Problem

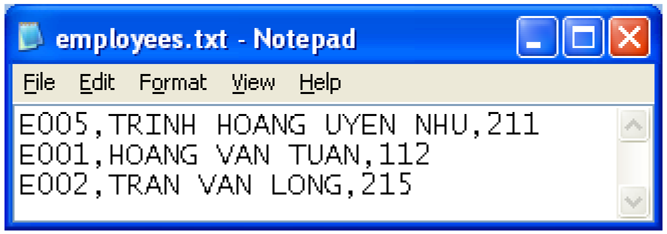
- Each employee details include: code, name, salary
- The text file, named employees.txt contains some initial employee details in the following line-by-line format

  code, name,salary
- Write a Java program having a simple menu that allows users managing a list of employees. Functions are supported:
  - Adding new employee
  - Removing employee.
  - Promoting the salary of an employee.
  - Listing employee details.
  - Save the list to file
  - Quit

Access Text Files …: Case study 1- Design



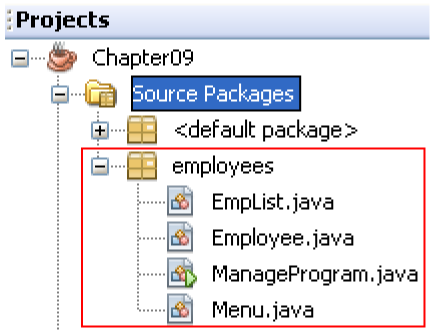
```
G:\GiangDay\FU\CoreJava\Chapter09
⊟ Chapter09            build
  ⊞ build              nbproject
  ⊞ nbproject          src
  ⊞ src                test
    test               build.xml
                       employees.txt
```



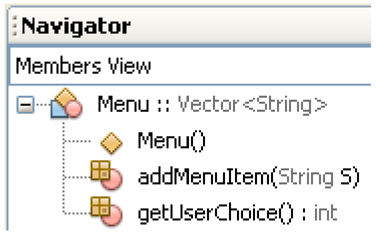
employees.txt - Notepad

File Edit Format View Help

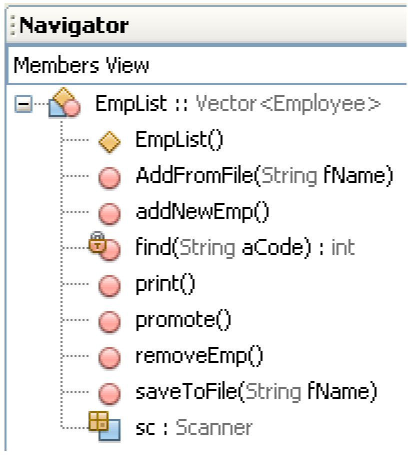
```
E005,TRINH HOANG UYEN NHU,211
E001,HOANG VAN TUAN,112
E002,TRAN VAN LONG,215
```



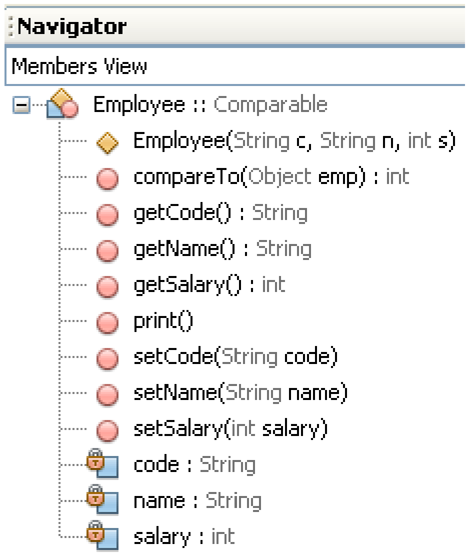
```
Projects
⊟ Chapter09
  ⊟ Source Packages
    ⊞ <default package>
    ⊟ employees
        EmpList.java
        Employee.java
        ManageProgram.java
        Menu.java
```



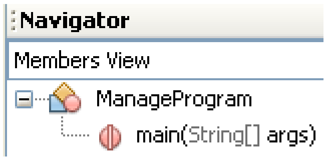
```
Navigator
Members View
⊟ Menu :: Vector<String>
    Menu()
    addMenuItem(String S)
    getUserChoice() : int
```



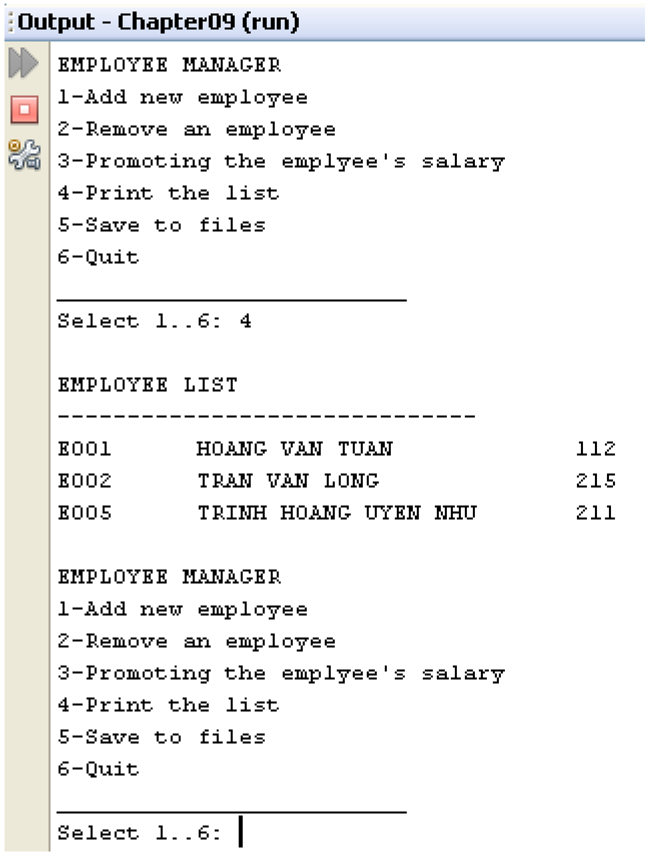
```
Navigator
Members View
⊟ EmpList :: Vector<Employee>
    EmpList()
    AddFromFile(String fName)
    addNewEmp()
    find(String aCode) : int
    print()
    promote()
    removeEmp()
    saveToFile(String fName)
    sc : Scanner
```



```
Navigator
Members View
⊟ Employee :: Comparable
    Employee(String c, String n, int s)
    compareTo(Object emp) : int
    getCode() : String
    getName() : String
    getSalary() : int
    print()
    setCode(String code)
    setName(String name)
    setSalary(int salary)
    code : String
    name : String
    salary : int
```



```
Navigator
Members View
⊟ ManageProgram
    main(String[] args)
```

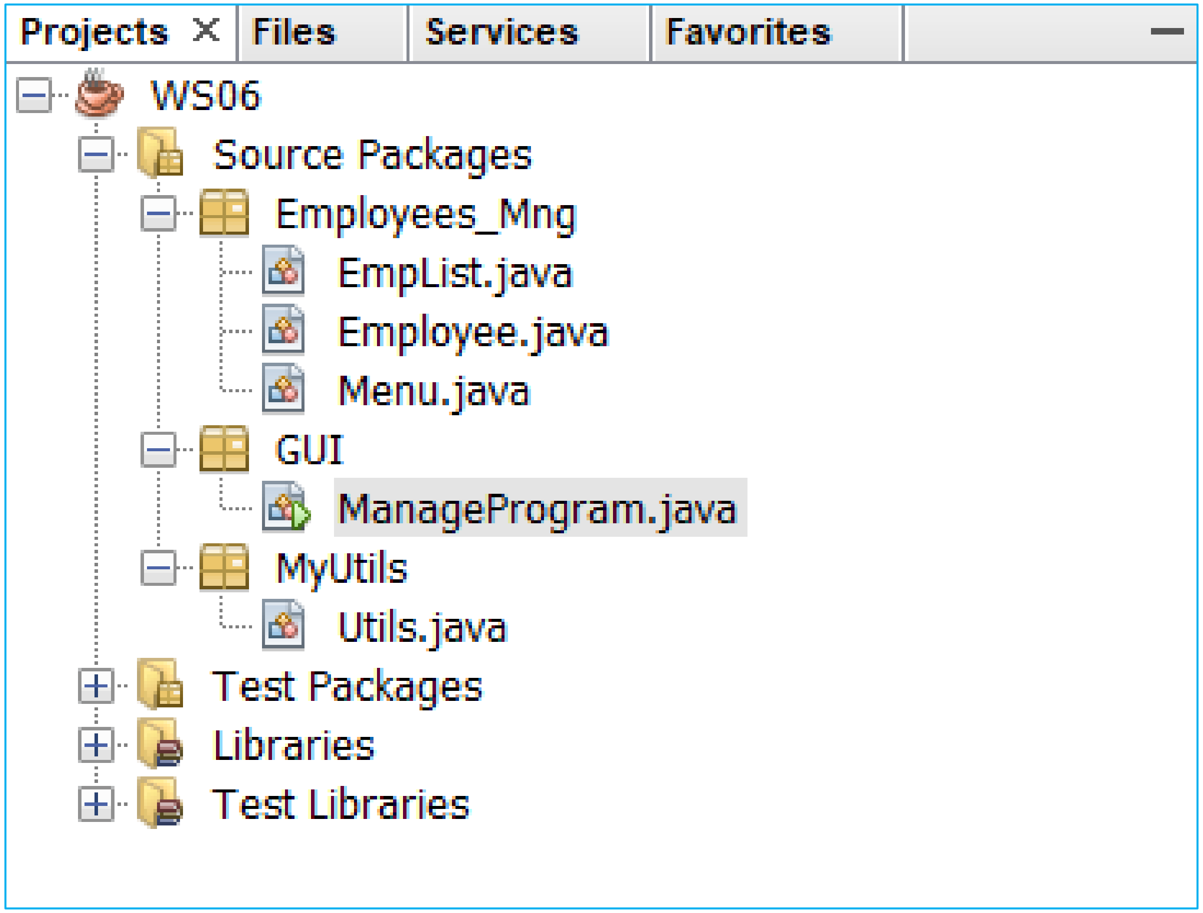


Output - Chapter09 (run)

```
EMPLOYEE MANAGER
1-Add new employee
2-Remove an employee
3-Promoting the emplyee's salary
4-Print the list
5-Save to files
6-Quit
-------------------------
Select 1..6: 4

EMPLOYEE LIST
-----------------------------
E001      HOANG VAN TUAN              112
E002      TRAN VAN LONG               215
E005      TRINH HOANG UYEN NHU        211

EMPLOYEE MANAGER
1-Add new employee
2-Remove an employee
3-Promoting the emplyee's salary
4-Print the list
5-Save to files
6-Quit
-------------------------
Select 1..6: 
```

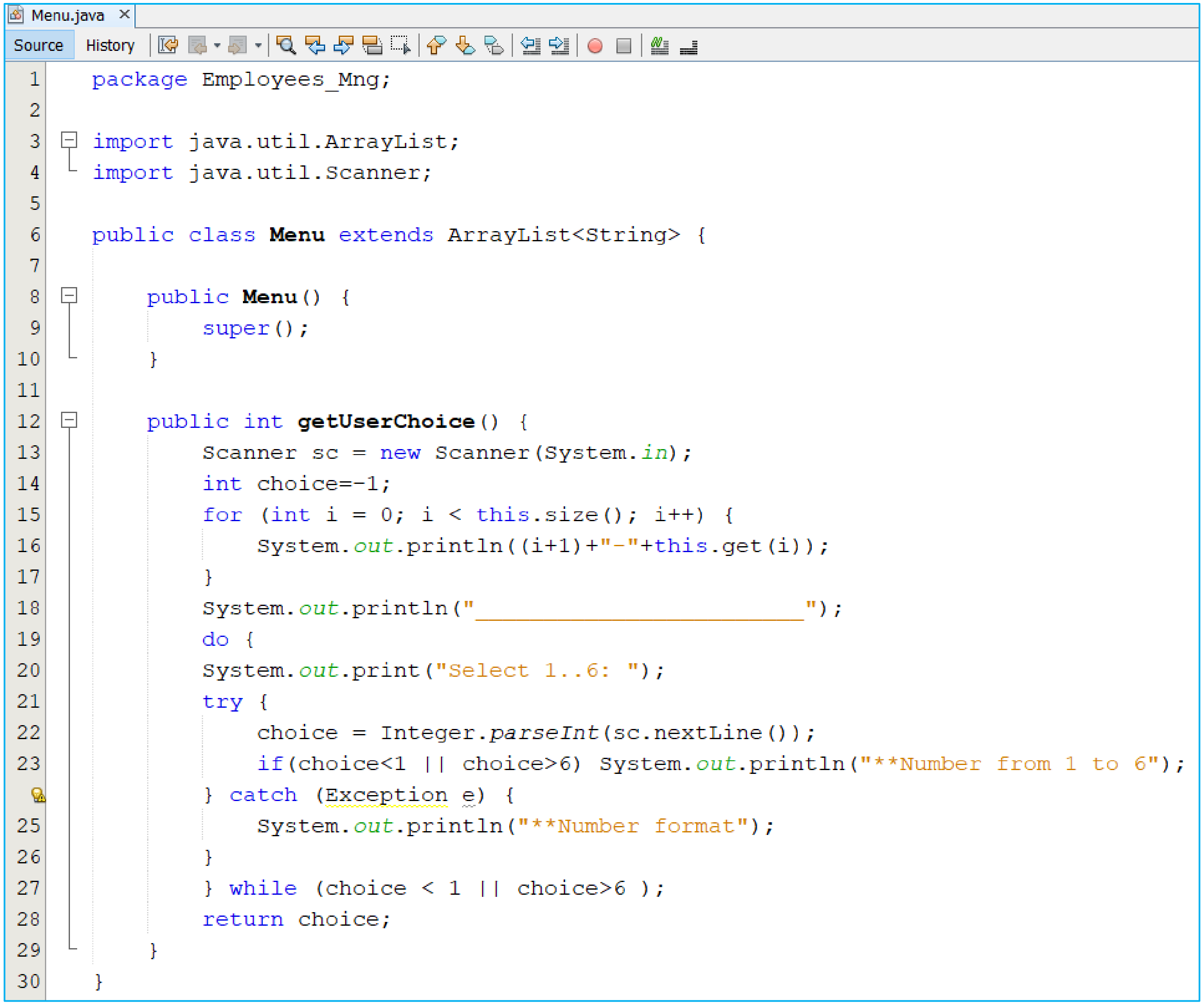
TRƯỜNG ĐẠI HỌC FPT

Access Text Files …: Case study 1- Implementations



```
WS06
  Source Packages
    Employees_Mng
      EmpList.java
      Employee.java
      Menu.java
    GUI
      ManageProgram.java
    MyUtils
      Utils.java
  Test Packages
  Libraries
  Test Libraries
```

Access Text Files …: Case study 1- Implementations

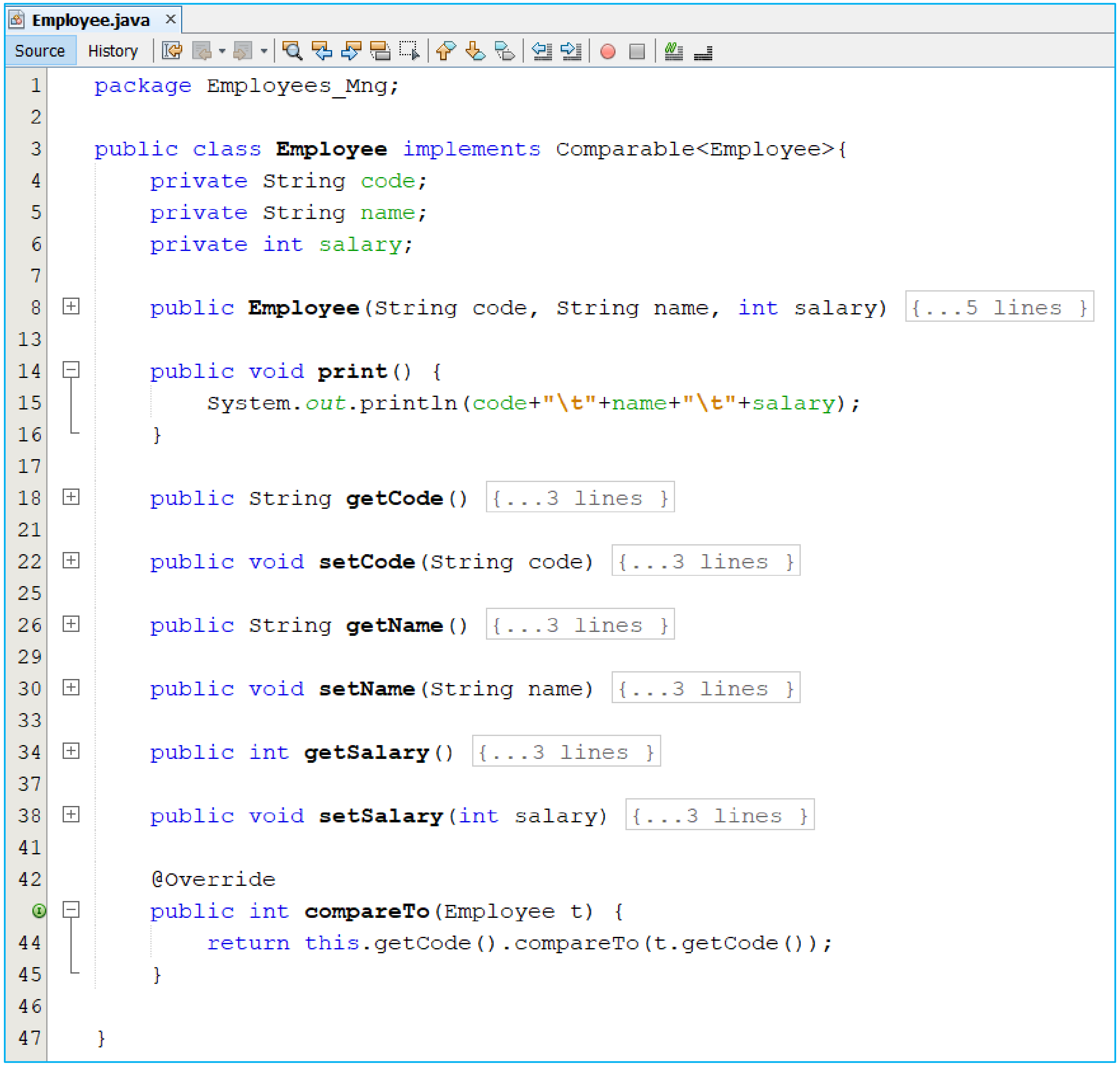


Menu.java

Source History

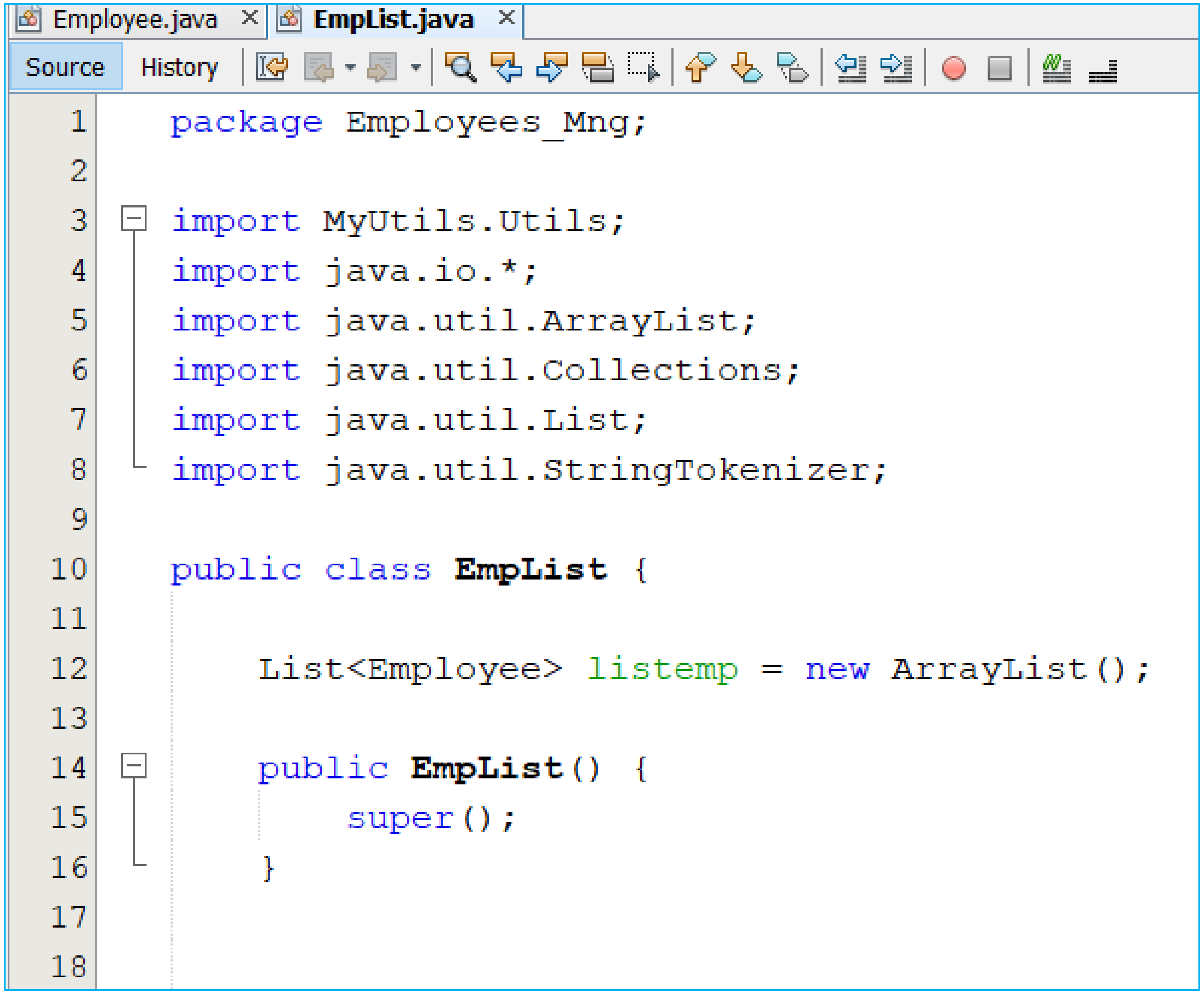
```
package Employees_Mng;

import java.util.ArrayList;
import java.util.Scanner;

public class Menu extends ArrayList<String> {

    public Menu() {
        super();
    }

    public int getUserChoice() {
        Scanner sc = new Scanner(System.in);
        int choice=-1;
        for (int i = 0; i < this.size(); i++) {
            System.out.println((i+1)+"-"+this.get(i));
        }
        System.out.println("________________________________");
        do {
        System.out.print("Select 1..6: ");
        try {
            choice = Integer.parseInt(sc.nextLine());
            if(choice<1 || choice>6) System.out.println("**Number from 1 to 6");
        } catch (Exception e) {
            System.out.println("**Number format");
        }
        } while (choice < 1 || choice>6 );
        return choice;
    }
}
```

Access Text Files …: Case study 1- Implementations



```
package Employees_Mng;

public class Employee implements Comparable<Employee>{
    private String code;
    private String name;
    private int salary;

    public Employee(String code, String name, int salary) {...5 lines }

    public void print() {
        System.out.println(code+"\t"+name+"\t"+salary);
    }

    public String getCode() {...3 lines }

    public void setCode(String code) {...3 lines }

    public String getName() {...3 lines }

    public void setName(String name) {...3 lines }

    public int getSalary() {...3 lines }

    public void setSalary(int salary) {...3 lines }

    @Override
    public int compareTo(Employee t) {
        return this.getCode().compareTo(t.getCode());
    }

}
```

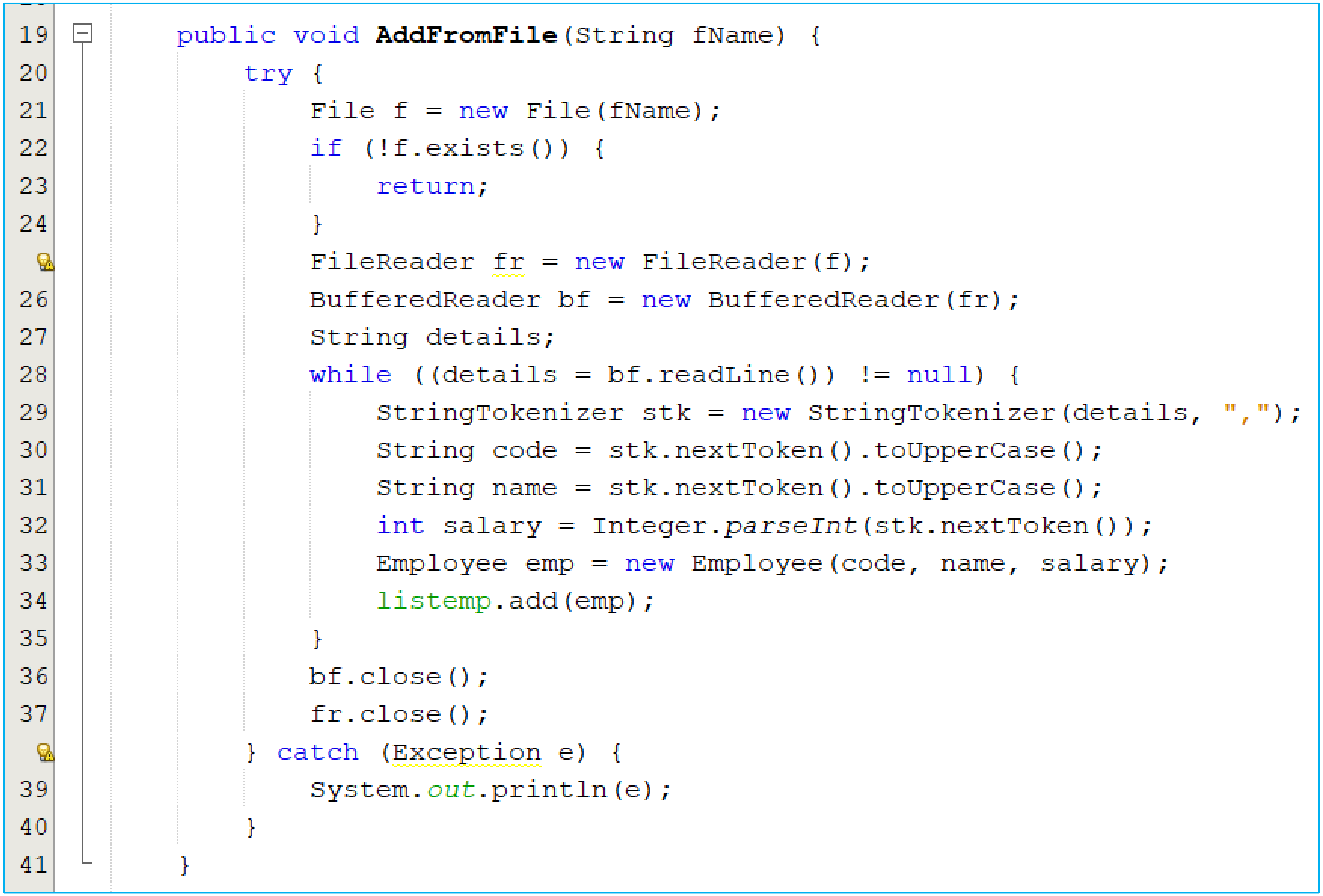
Access Text Files …: Case study 1- Implementations



```
package Employees_Mng;

import MyUtils.Utils;
import java.io.*;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.StringTokenizer;

public class EmpList {

    List<Employee> listemp = new ArrayList();

    public EmpList() {
        super();
    }


```

TRƯỜNG ĐẠI HỌC FPT

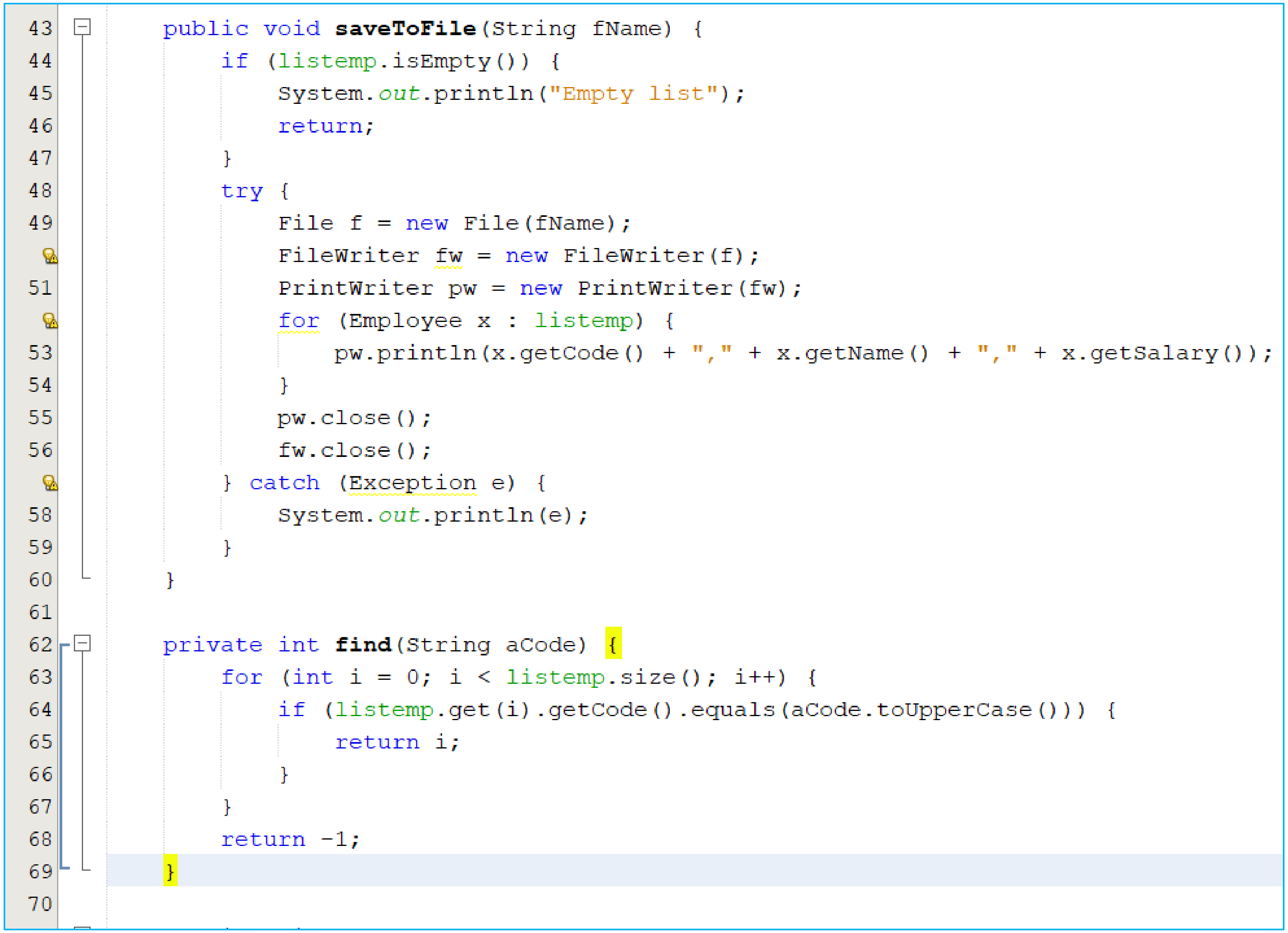
Access Text Files …: Case study 1- Implementations



```java
    public void AddFromFile(String fName) {
        try {
            File f = new File(fName);
            if (!f.exists()) {
                return;
            }
            FileReader fr = new FileReader(f);
            BufferedReader bf = new BufferedReader(fr);
            String details;
            while ((details = bf.readLine()) != null) {
                StringTokenizer stk = new StringTokenizer(details, ",");
                String code = stk.nextToken().toUpperCase();
                String name = stk.nextToken().toUpperCase();
                int salary = Integer.parseInt(stk.nextToken());
                Employee emp = new Employee(code, name, salary);
                listemp.add(emp);
            }
            bf.close();
            fr.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
```

TRƯỜNG ĐẠI HỌC FPT

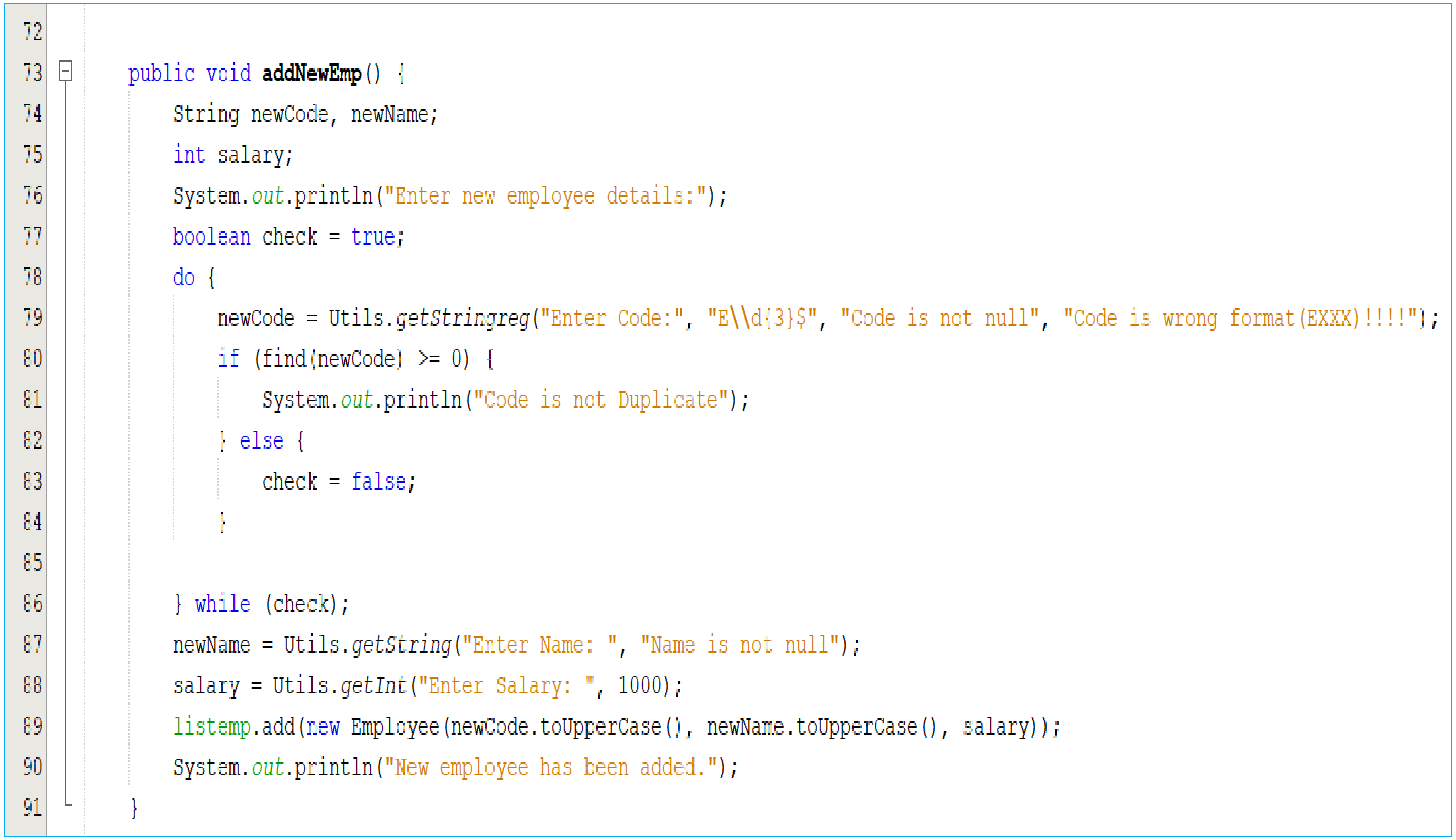
Access Text Files …: Case study 1- Implementations



```
    public void saveToFile(String fName) {
        if (listemp.isEmpty()) {
            System.out.println("Empty list");
            return;
        }
        try {
            File f = new File(fName);
            FileWriter fw = new FileWriter(f);
            PrintWriter pw = new PrintWriter(fw);
            for (Employee x : listemp) {
                pw.println(x.getCode() + "," + x.getName() + "," + x.getSalary());
            }
            pw.close();
            fw.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }

    private int find(String aCode) {
        for (int i = 0; i < listemp.size(); i++) {
            if (listemp.get(i).getCode().equals(aCode.toUpperCase())) {
                return i;
            }
        }
        return -1;
    }
```

TRƯỜNG ĐẠI HỌC FPT

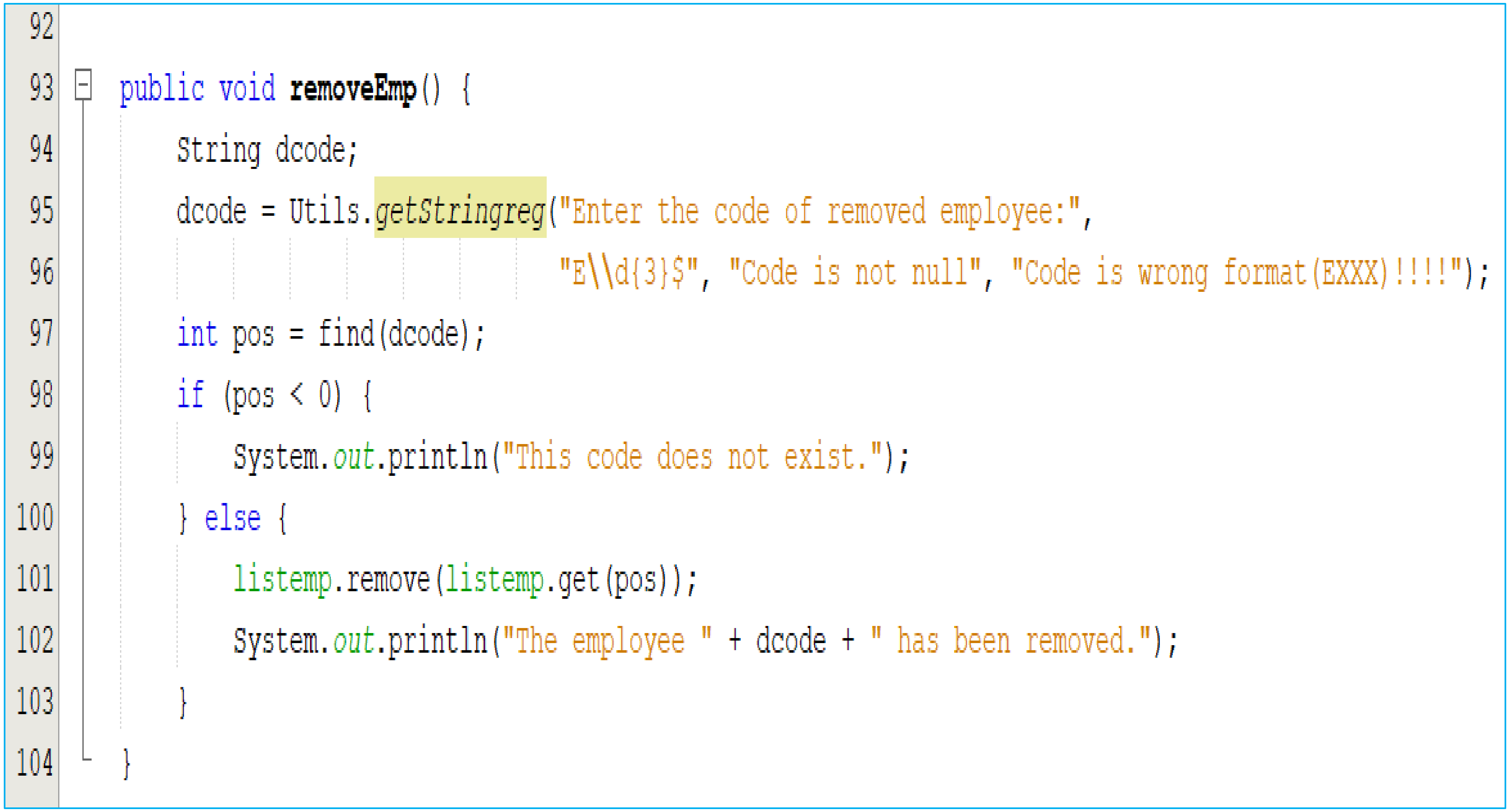
Access Text Files …: Case study 1- Implementations



```java

    public void addNewEmp() {
        String newCode, newName;
        int salary;
        System.out.println("Enter new employee details:");
        boolean check = true;
        do {
            newCode = Utils.getStringreg("Enter Code:", "E\\d{3}$", "Code is not null", "Code is wrong format(EXXX)!!!!");
            if (find(newCode) >= 0) {
                System.out.println("Code is not Duplicate");
            } else {
                check = false;
            }

        } while (check);
        newName = Utils.getString("Enter Name: ", "Name is not null");
        salary = Utils.getInt("Enter Salary: ", 1000);
        listemp.add(new Employee(newCode.toUpperCase(), newName.toUpperCase(), salary));
        System.out.println("New employee has been added.");
    }
```

TRƯỜNG ĐẠI HỌC FPT

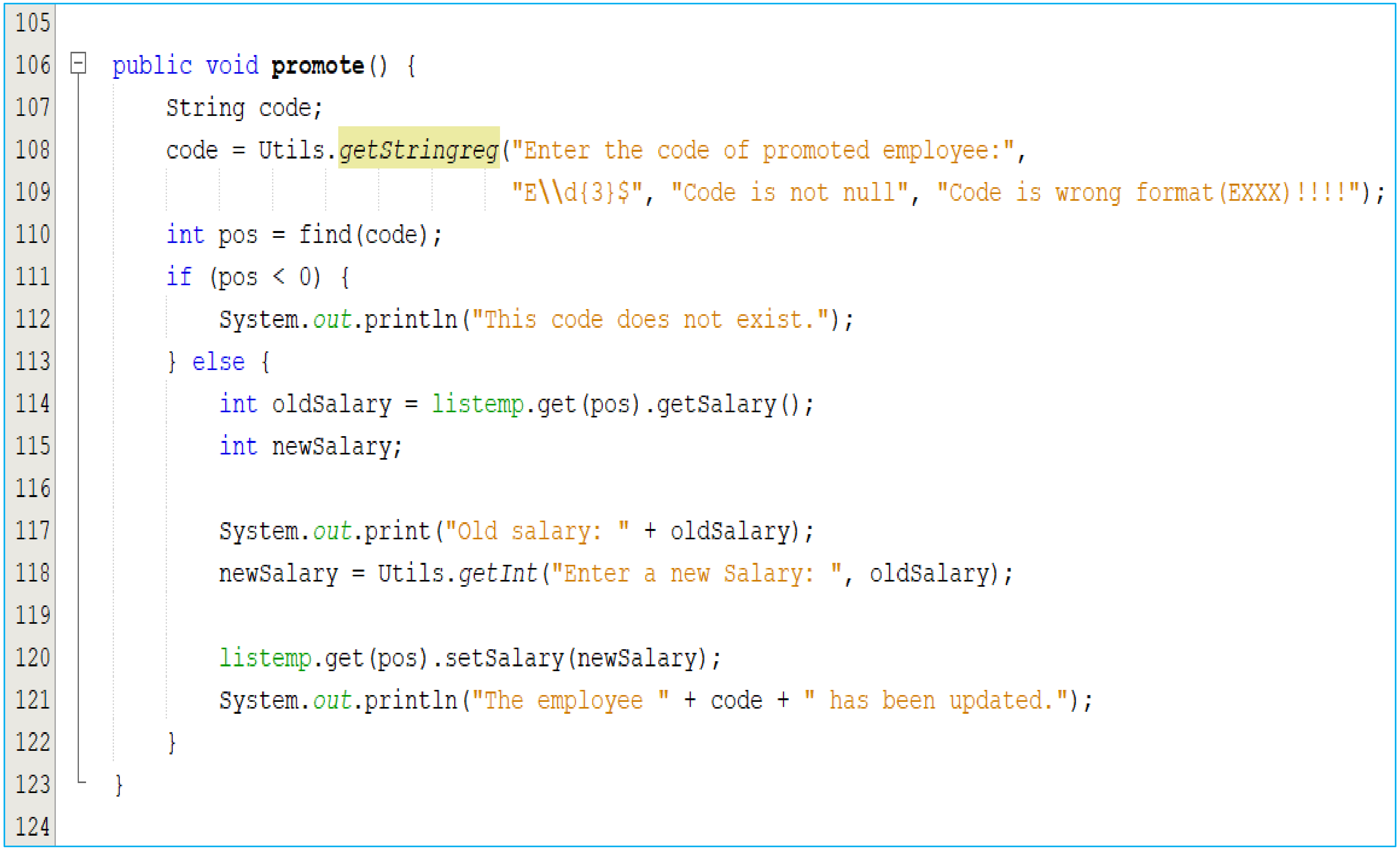
Access Text Files …: Case study 1- Implementations



```
public void removeEmp() {
    String dcode;
    dcode = Utils.getStringreg("Enter the code of removed employee:",
                        "E\\d{3}$", "Code is not null", "Code is wrong format(EXXX)!!!!");
    int pos = find(dcode);
    if (pos < 0) {
        System.out.println("This code does not exist.");
    } else {
        listemp.remove(listemp.get(pos));
        System.out.println("The employee " + dcode + " has been removed.");
    }
}
```

TRƯỜNG ĐẠI HỌC FPT

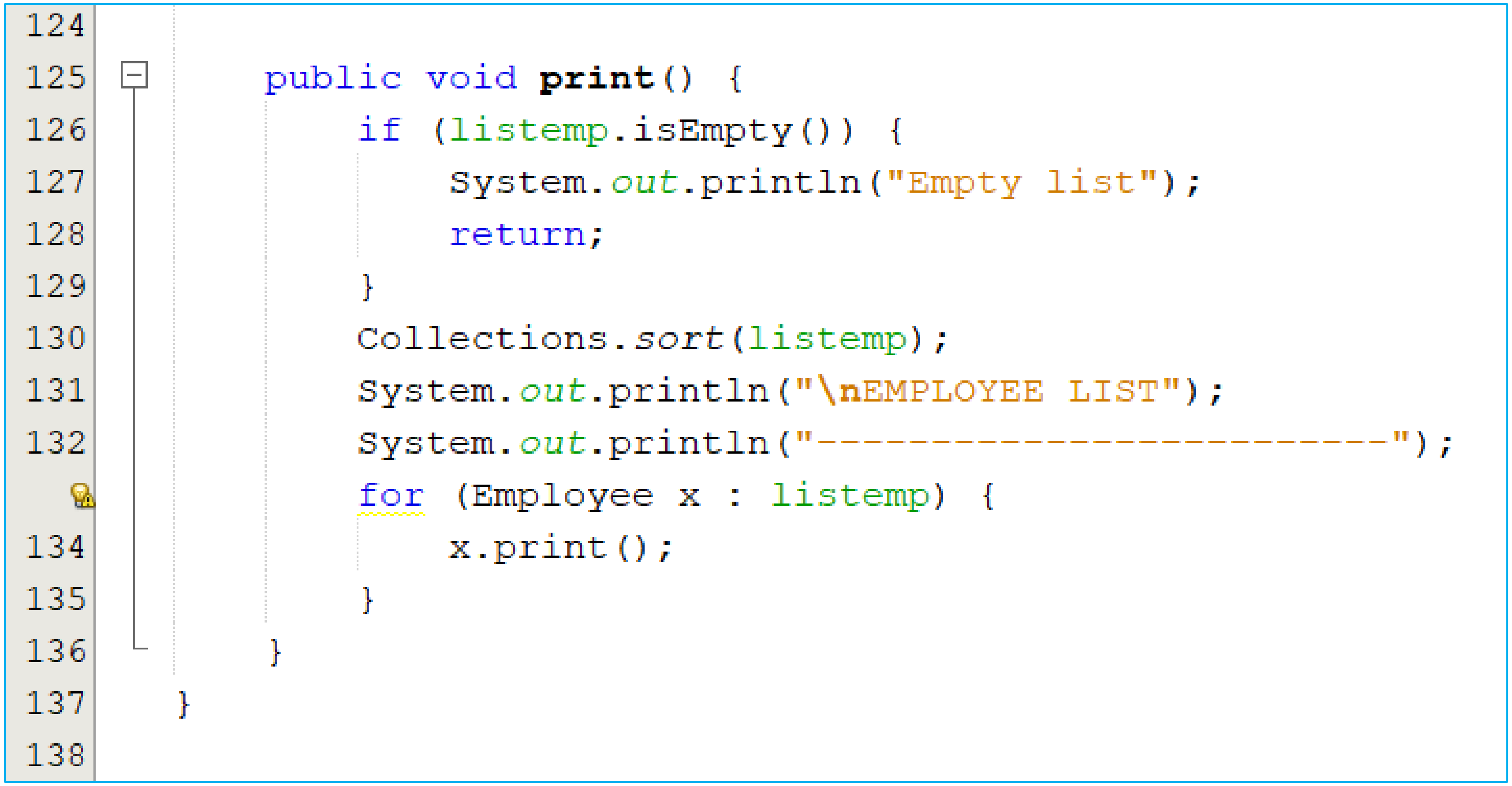
Access Text Files …: Case study 1- Implementations



```java
public void promote() {
    String code;
    code = Utils.getStringreg("Enter the code of promoted employee:",
                             "E\\d{3}$", "Code is not null", "Code is wrong format(EXXX)!!!!");
    int pos = find(code);
    if (pos < 0) {
        System.out.println("This code does not exist.");
    } else {
        int oldSalary = listemp.get(pos).getSalary();
        int newSalary;

        System.out.print("Old salary: " + oldSalary);
        newSalary = Utils.getInt("Enter a new Salary: ", oldSalary);

        listemp.get(pos).setSalary(newSalary);
        System.out.println("The employee " + code + " has been updated.");
    }
}
```

TRƯỜNG ĐẠI HỌC FPT

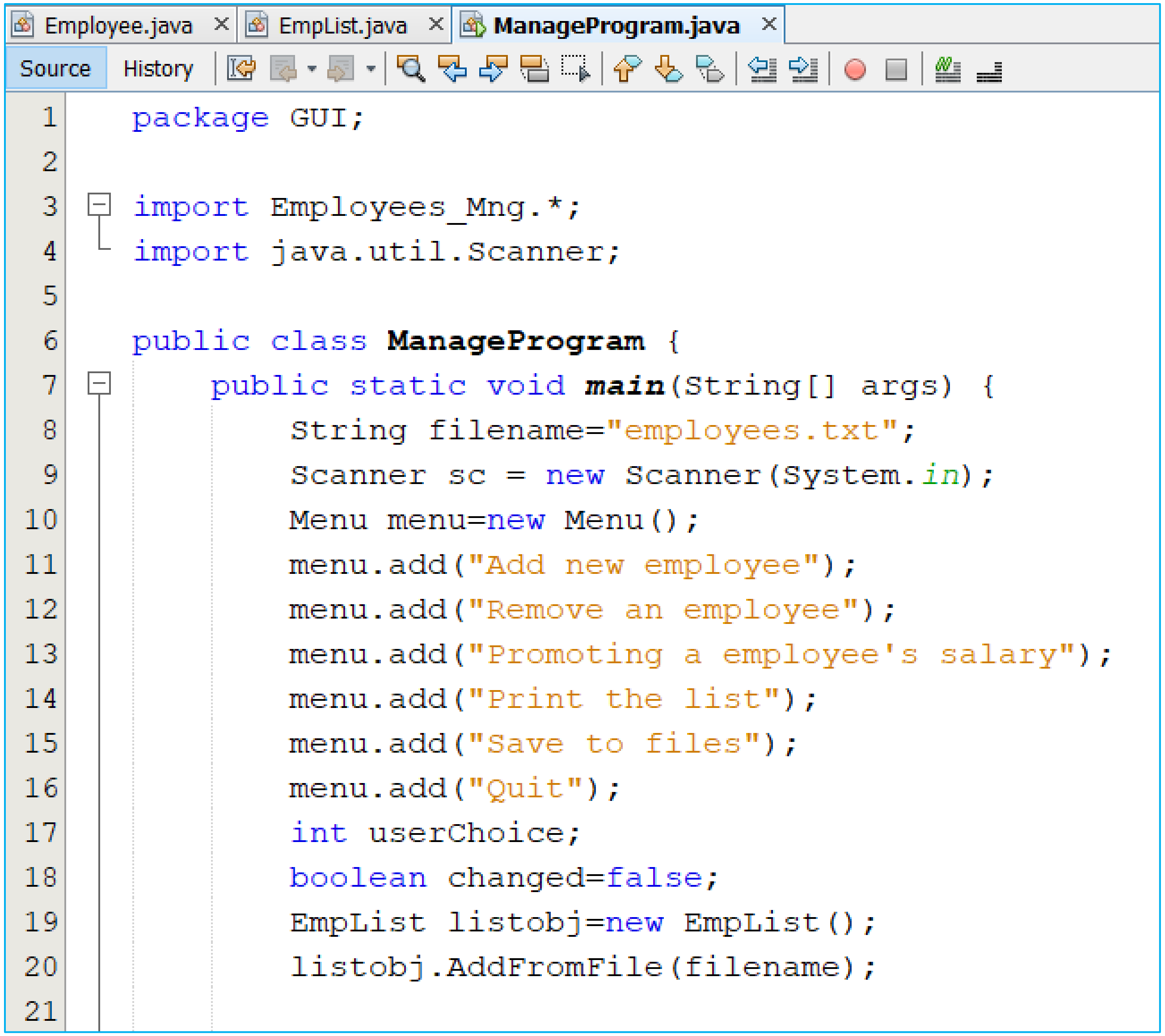
Access Text Files …: Case study 1- Implementations



```
    public void print() {
        if (listemp.isEmpty()) {
            System.out.println("Empty list");
            return;
        }
        Collections.sort(listemp);
        System.out.println("\nEMPLOYEE LIST");
        System.out.println("-------------------------");
        for (Employee x : listemp) {
            x.print();
        }
    }
}
```

TRƯỜNG ĐẠI HỌC FPT

Access Text Files …: Case study 1- Implementations



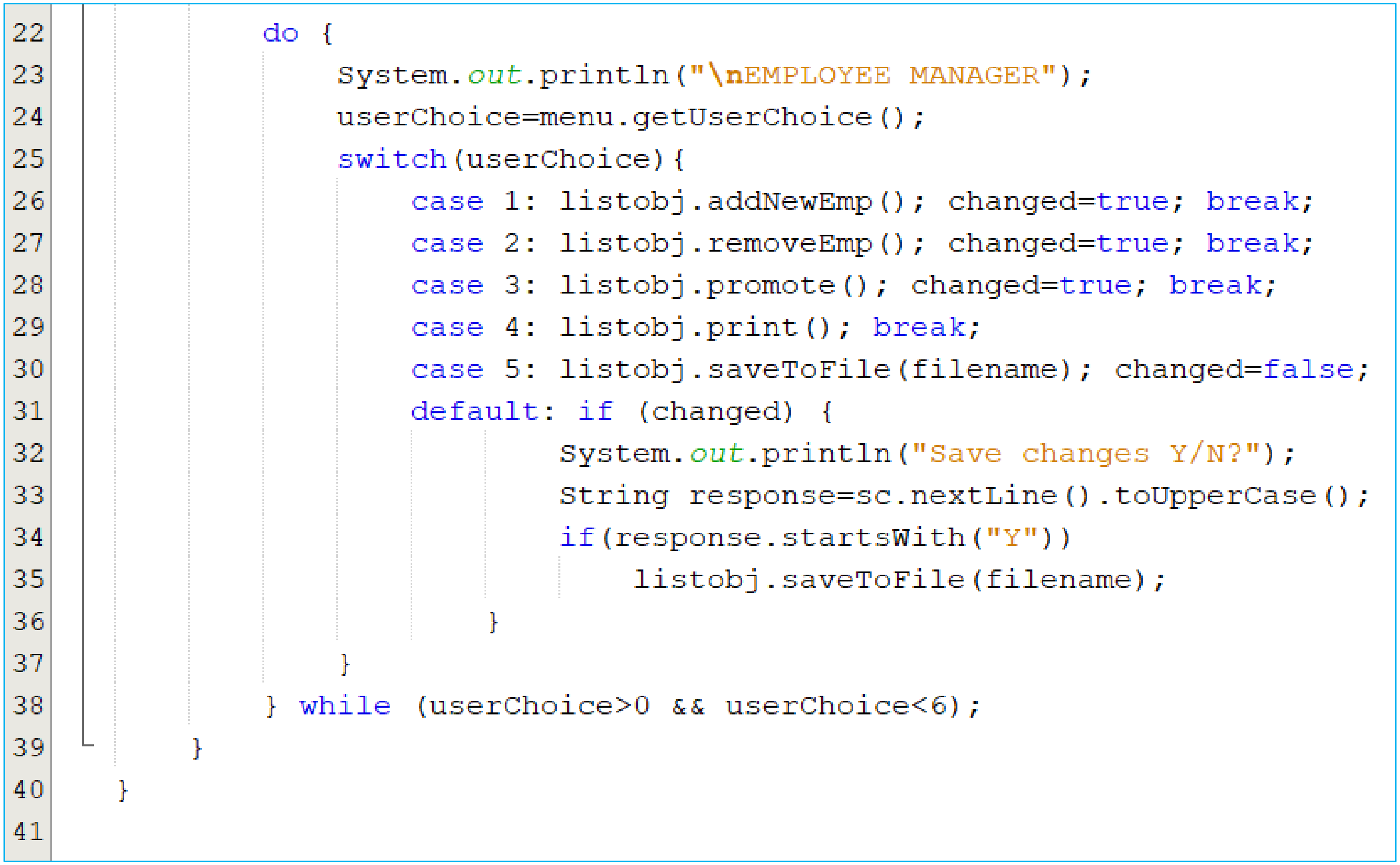
Employee.java EmpList.java ManageProgram.java

Source History

```
package GUI;

import Employees_Mng.*;
import java.util.Scanner;

public class ManageProgram {
    public static void main(String[] args) {
        String filename="employees.txt";
        Scanner sc = new Scanner(System.in);
        Menu menu=new Menu();
        menu.add("Add new employee");
        menu.add("Remove an employee");
        menu.add("Promoting a employee's salary");
        menu.add("Print the list");
        menu.add("Save to files");
        menu.add("Quit");
        int userChoice;
        boolean changed=false;
        EmpList listobj=new EmpList();
        listobj.AddFromFile(filename);

```

TRƯỜNG ĐẠI HỌC FPT

Access Text Files …: Case study 1- Implementations

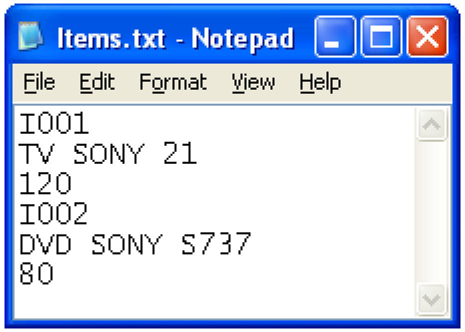


```
            do {
                System.out.println("\nEMPLOYEE MANAGER");
                userChoice=menu.getUserChoice();
                switch(userChoice){
                    case 1: listobj.addNewEmp(); changed=true; break;
                    case 2: listobj.removeEmp(); changed=true; break;
                    case 3: listobj.promote(); changed=true; break;
                    case 4: listobj.print(); break;
                    case 5: listobj.saveToFile(filename); changed=false;
                    default: if (changed) {
                                System.out.println("Save changes Y/N?");
                                String response=sc.nextLine().toUpperCase();
                                if(response.startsWith("Y"))
                                    listobj.saveToFile(filename);
                        }
                }
            } while (userChoice>0 && userChoice<6);
        }
    }
```

Access Text Files …: Case study 2.- Append File Demo.

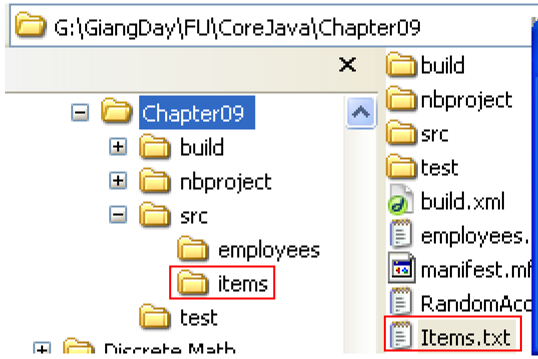
Problem

- Each item details include: code, name, price. The item's code can not be duplicated.
- An accountant can not be allowed to view all stored items (in the text file, named items.txt) but he/she can add some new items to this file.
- Data format in this file (line by line):
  - Line for the code of item
  - Line for the name of item
  - Line for the price of item
- Write a Java program having a simple menu which allows users managing a item list through program's functions:
  - Add new item
  - Update an item
  - Delete an item
  - Save items(Appending items to this file)



```
I001
TV SONY 21
120
I002
DVD SONY S737
80
```

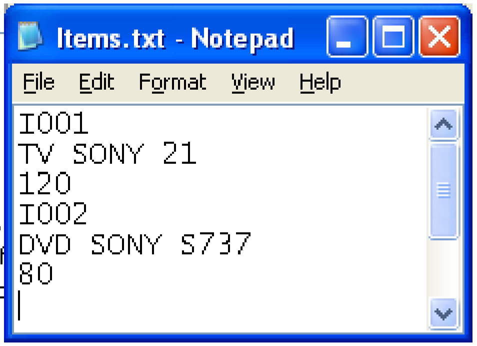
Access Text Files …: Case study 2.-Design



G:\GiangDay\FU\CoreJava\Chapter09

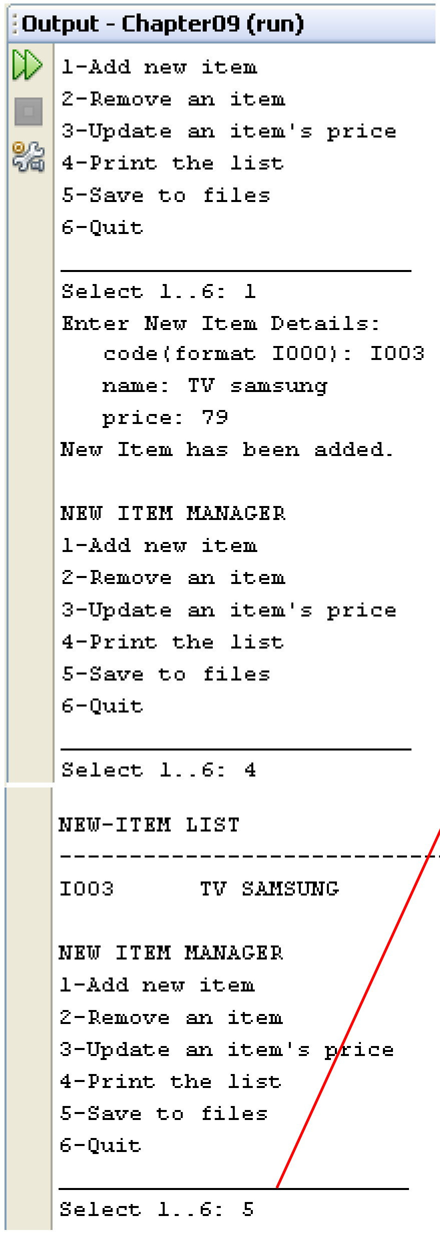
```
Chapter09
  build
  nbproject
  src
    employees
    items
  test
Discrete Math
```

```
build
nbproject
src
test
build.xml
employees.
manifest.mf
RandomAcc
Items.txt
```



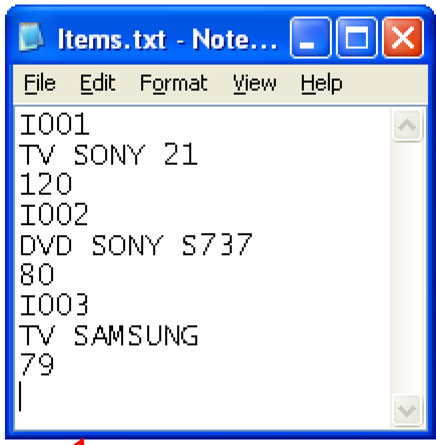
Items.txt - Notepad

```
I001
TV  SONY  21
120
I002
DVD  SONY  S737
80
```



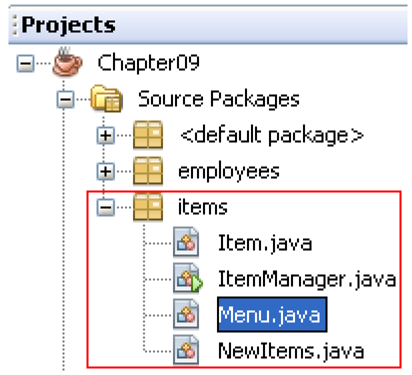
Output - Chapter09 (run)

```
1-Add new item
2-Remove an item
3-Update an item's price
4-Print the list
5-Save to files
6-Quit
-----------------------------------
Select 1..6: 1
Enter New Item Details:
    code(format I000): I003
    name: TV samsung
    price: 79
New Item has been added.

NEW ITEM MANAGER
1-Add new item
2-Remove an item
3-Update an item's price
4-Print the list
5-Save to files
6-Quit
-----------------------------------
Select 1..6: 4

NEW-ITEM LIST
----------------------------------------
I003       TV SAMSUNG                 79

NEW ITEM MANAGER
1-Add new item
2-Remove an item
3-Update an item's price
4-Print the list
5-Save to files
6-Quit
-----------------------------------
Select 1..6: 5
```



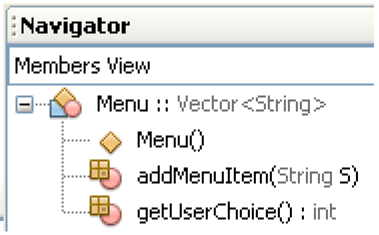
Items.txt - Note...

```
I001
TV  SONY  21
120
I002
DVD  SONY  S737
80
I003
TV  SAMSUNG
79
```



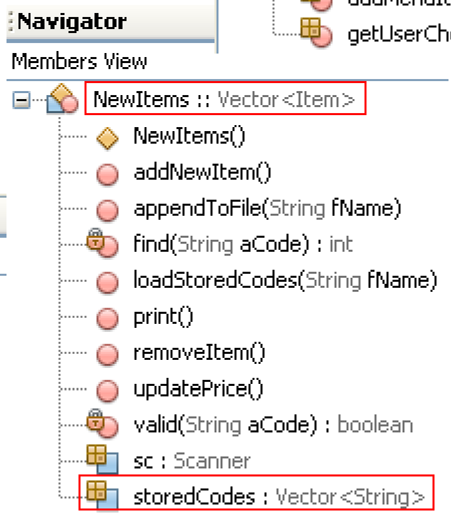
Projects

```
Chapter09
  Source Packages
    <default package>
    employees
    items
      Item.java
      ItemManager.java
      Menu.java
      NewItems.java
```



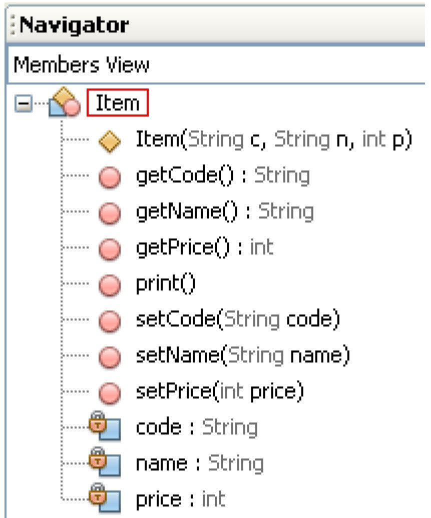
Navigator — Members View

```
Menu :: Vector<String>
  Menu()
  addMenuItem(String S)
  getUserChoice() : int
```



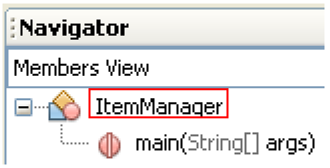
Navigator — Members View

```
NewItems :: Vector<Item>
  NewItems()
  addNewItem()
  appendToFile(String fName)
  find(String aCode) : int
  loadStoredCodes(String fName)
  print()
  removeItem()
  updatePrice()
  valid(String aCode) : boolean
  sc : Scanner
  storedCodes : Vector<String>
```



Navigator — Members View

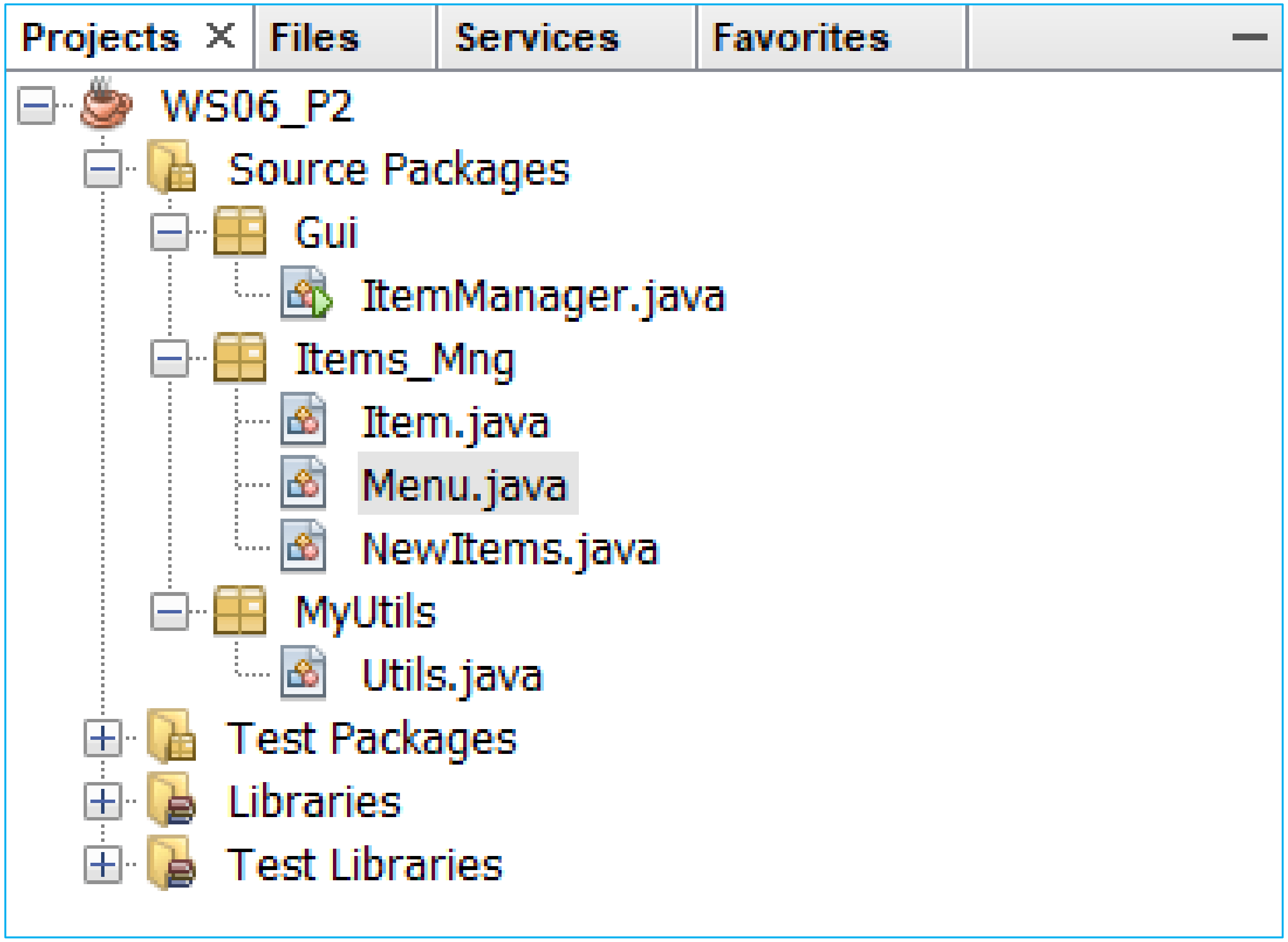
```
Item
  Item(String c, String n, int p)
  getCode() : String
  getName() : String
  getPrice() : int
  print()
  setCode(String code)
  setName(String name)
  setPrice(int price)
  code : String
  name : String
  price : int
```



Navigator — Members View

```
ItemManager
  main(String[] args)
```

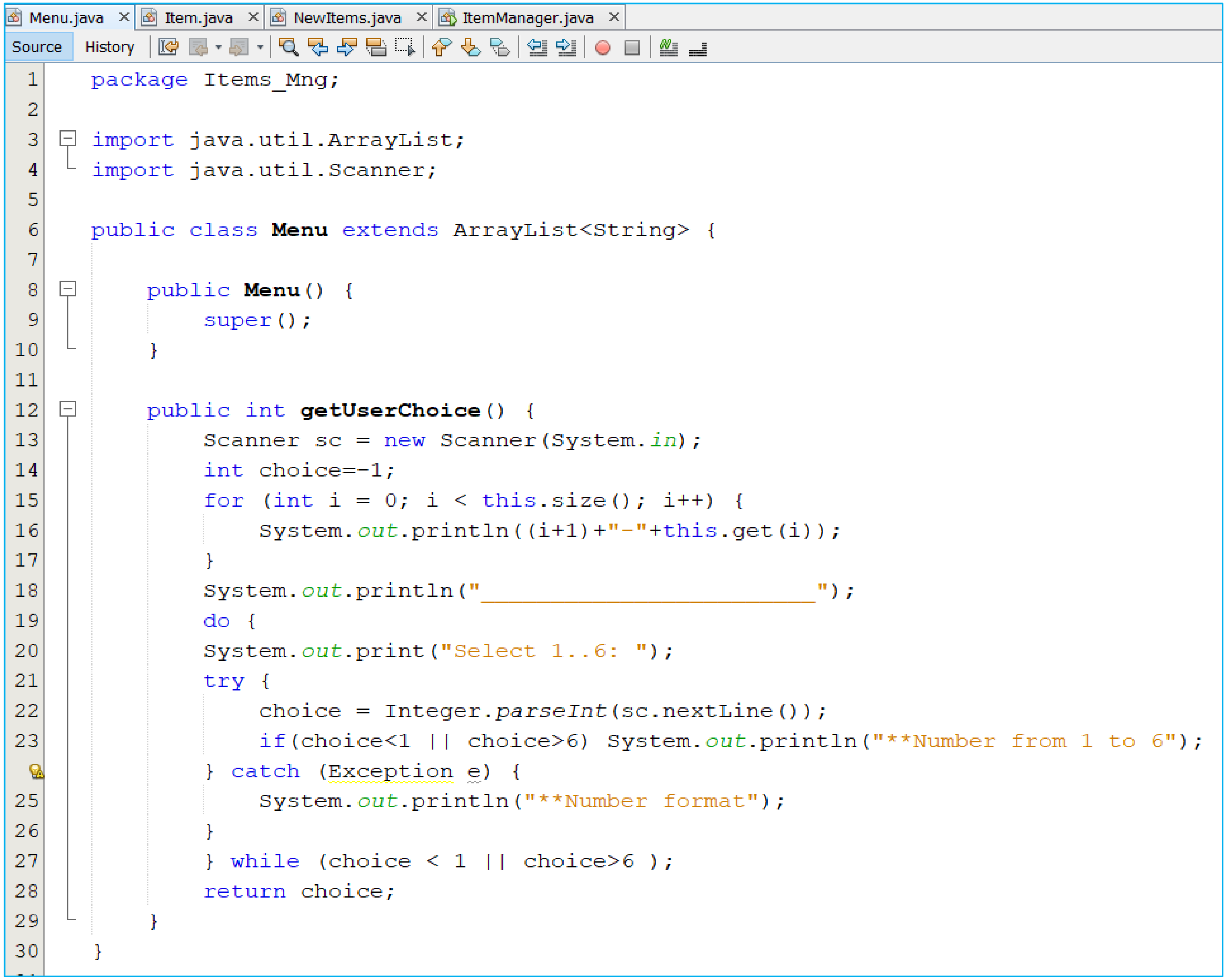
Access Text Files …: Case study 2- Implementations



```
WS06_P2
    Source Packages
        Gui
            ItemManager.java
        Items_Mng
            Item.java
            Menu.java
            NewItems.java
        MyUtils
            Utils.java
    Test Packages
    Libraries
    Test Libraries
```

TRƯỜNG ĐẠI HỌC FPT

Access Text Files …: Case study 4.- Implementations



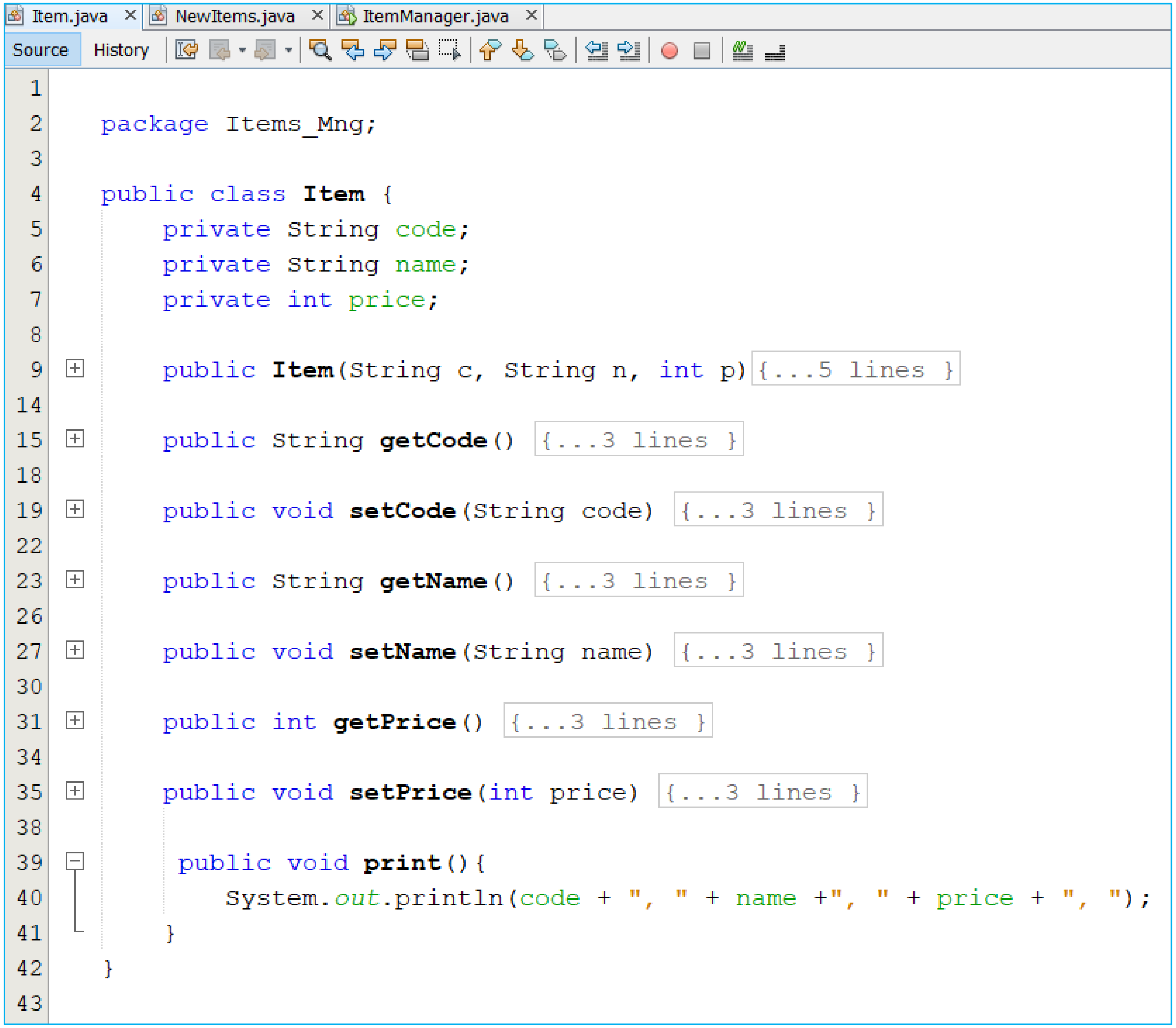
Menu.java | Item.java | NewItems.java | ItemManager.java

Source | History

```java
package Items_Mng;

import java.util.ArrayList;
import java.util.Scanner;

public class Menu extends ArrayList<String> {

    public Menu() {
        super();
    }

    public int getUserChoice() {
        Scanner sc = new Scanner(System.in);
        int choice=-1;
        for (int i = 0; i < this.size(); i++) {
            System.out.println((i+1)+"-"+this.get(i));
        }
        System.out.println("________________________");
        do {
        System.out.print("Select 1..6: ");
        try {
            choice = Integer.parseInt(sc.nextLine());
            if(choice<1 || choice>6) System.out.println("**Number from 1 to 6");
        } catch (Exception e) {
            System.out.println("**Number format");
        }
        } while (choice < 1 || choice>6 );
        return choice;
    }
}
```

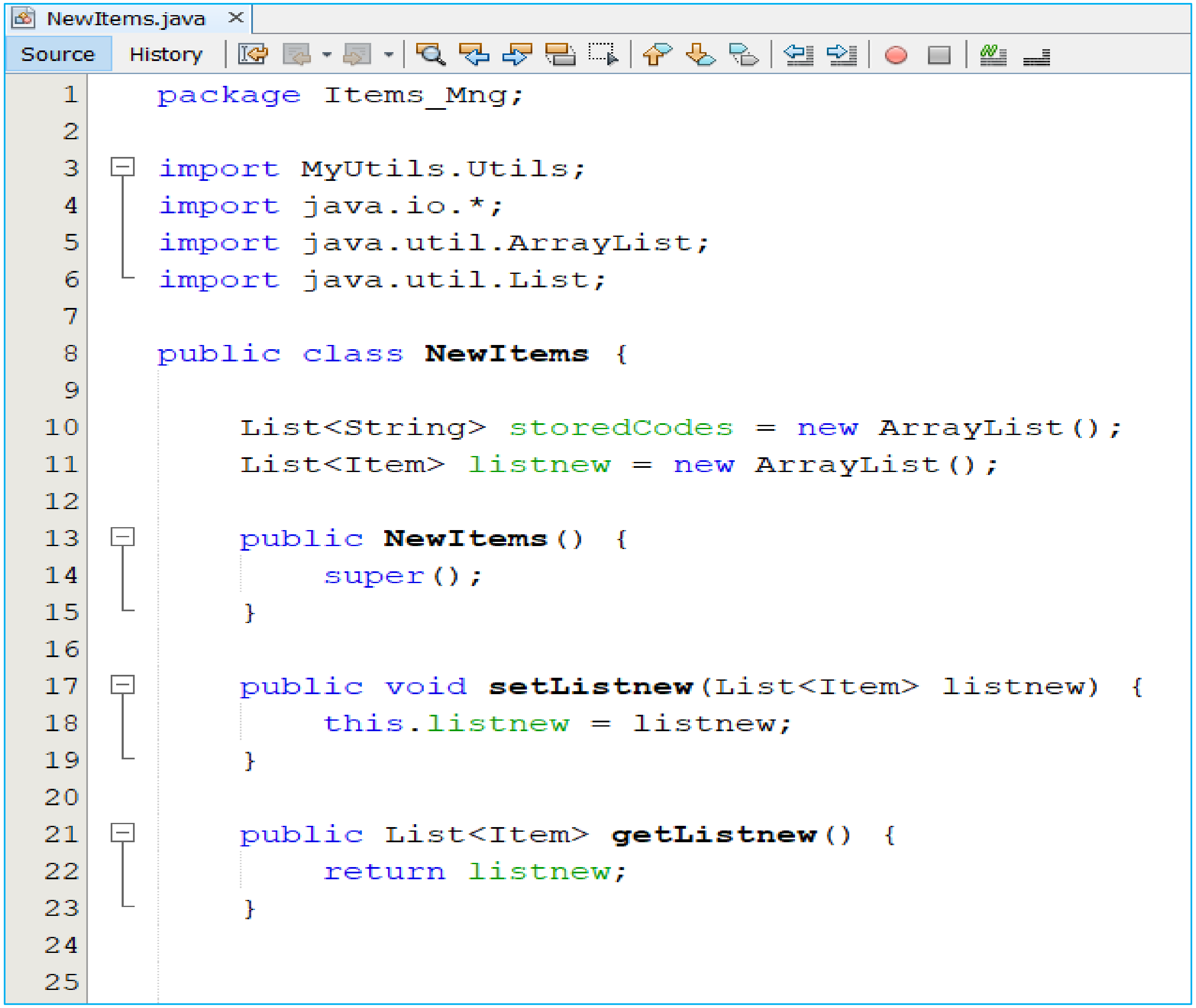
TRƯỜNG ĐẠI HỌC FPT

Access Text Files …: Case study 2- Implementations



```java

package Items_Mng;

public class Item {
    private String code;
    private String name;
    private int price;

    public Item(String c, String n, int p){...5 lines }

    public String getCode() {...3 lines }

    public void setCode(String code) {...3 lines }

    public String getName() {...3 lines }

    public void setName(String name) {...3 lines }

    public int getPrice() {...3 lines }

    public void setPrice(int price) {...3 lines }

     public void print(){
        System.out.println(code + ", " + name +", " + price + ", ");
    }
}

```

Access Text Files …: Case study 2- Implementations


NewItems.java

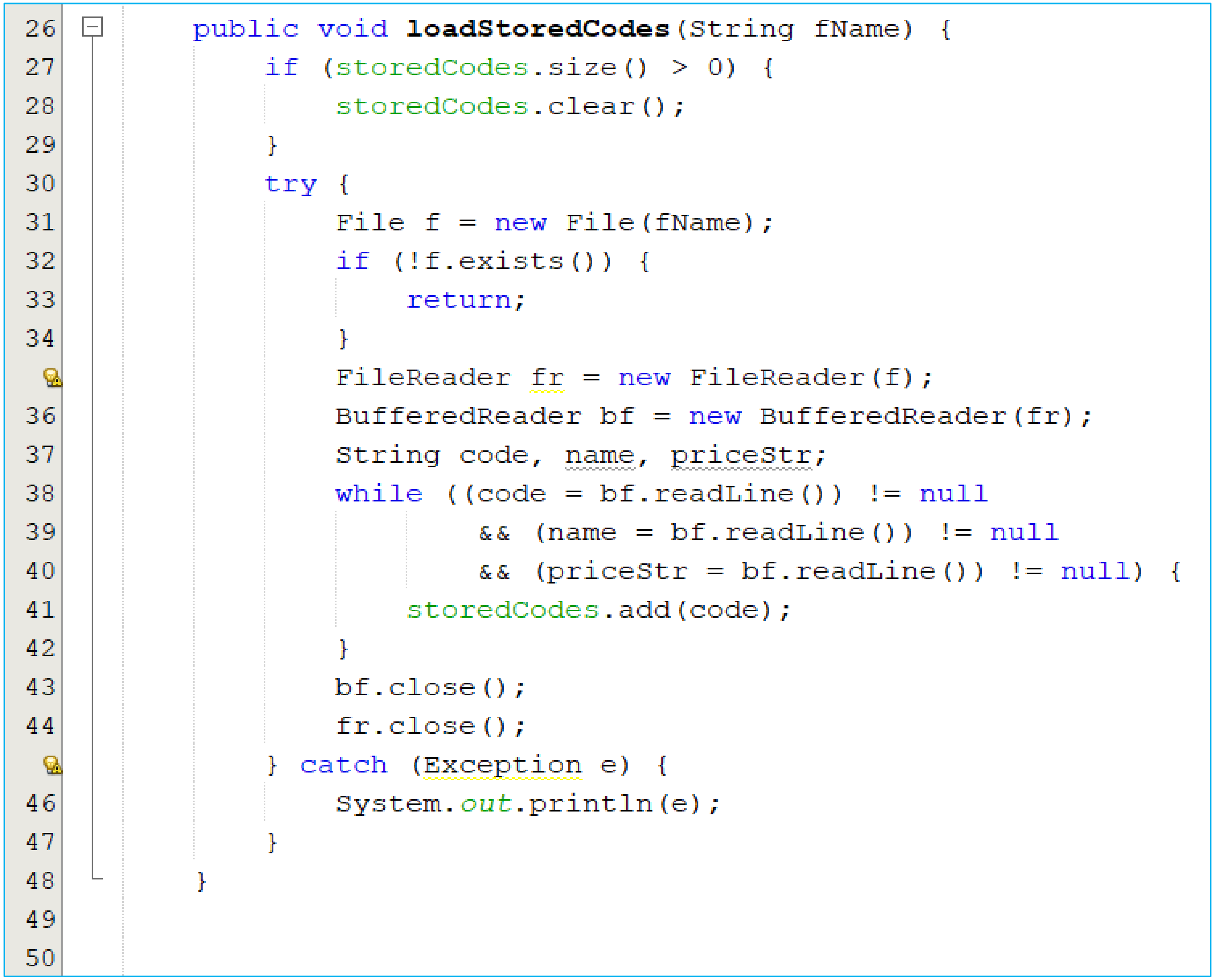
Source History

```
package Items_Mng;

import MyUtils.Utils;
import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class NewItems {

    List<String> storedCodes = new ArrayList();
    List<Item> listnew = new ArrayList();

    public NewItems() {
        super();
    }

    public void setListnew(List<Item> listnew) {
        this.listnew = listnew;
    }

    public List<Item> getListnew() {
        return listnew;
    }


```

Fpt University

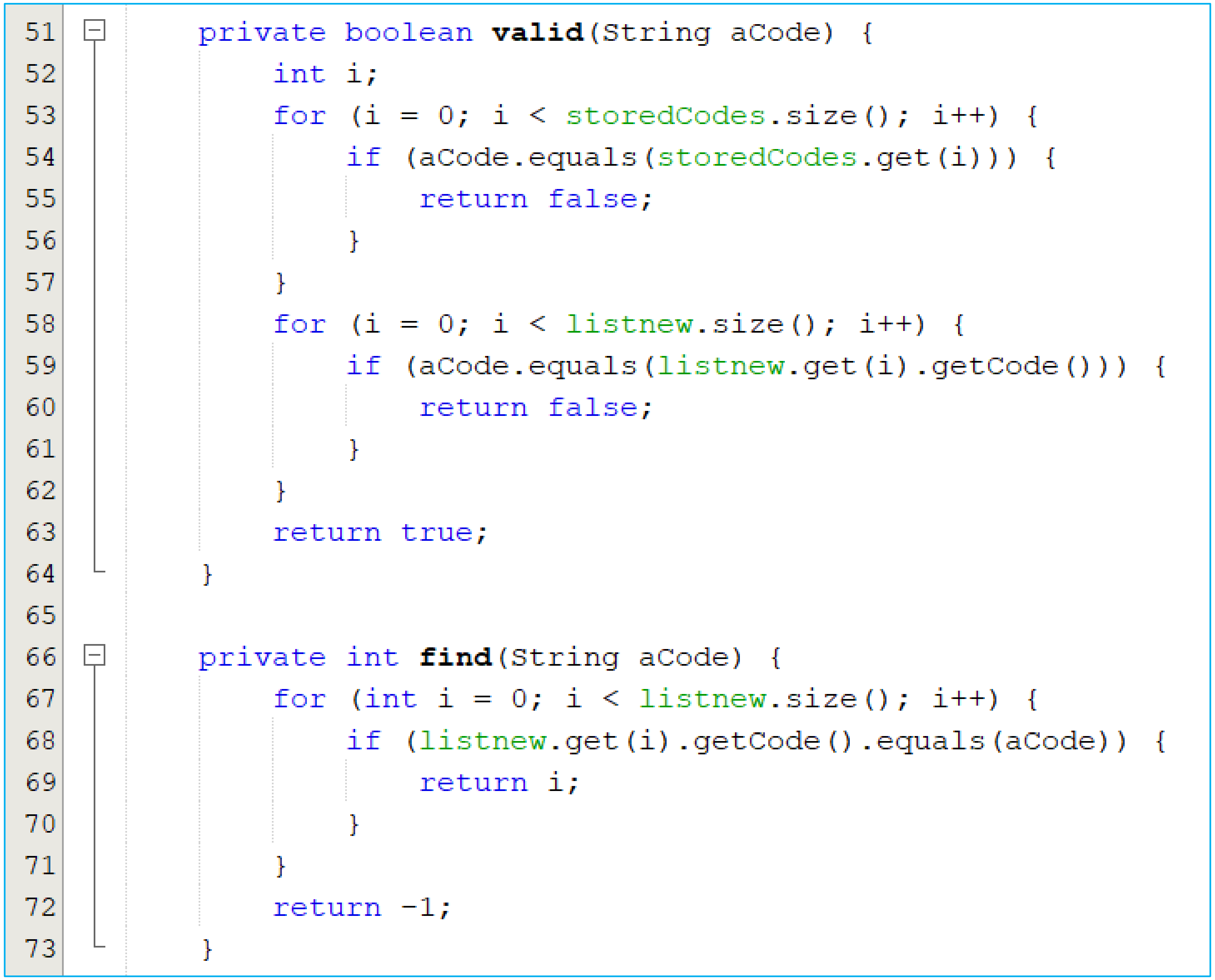
TRƯỜNG ĐẠI HỌC FPT

Access Text Files …: Case study 2- Implementations



```java
    public void loadStoredCodes(String fName) {
        if (storedCodes.size() > 0) {
            storedCodes.clear();
        }
        try {
            File f = new File(fName);
            if (!f.exists()) {
                return;
            }
            FileReader fr = new FileReader(f);
            BufferedReader bf = new BufferedReader(fr);
            String code, name, priceStr;
            while ((code = bf.readLine()) != null
                    && (name = bf.readLine()) != null
                    && (priceStr = bf.readLine()) != null) {
                storedCodes.add(code);
            }
            bf.close();
            fr.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
```

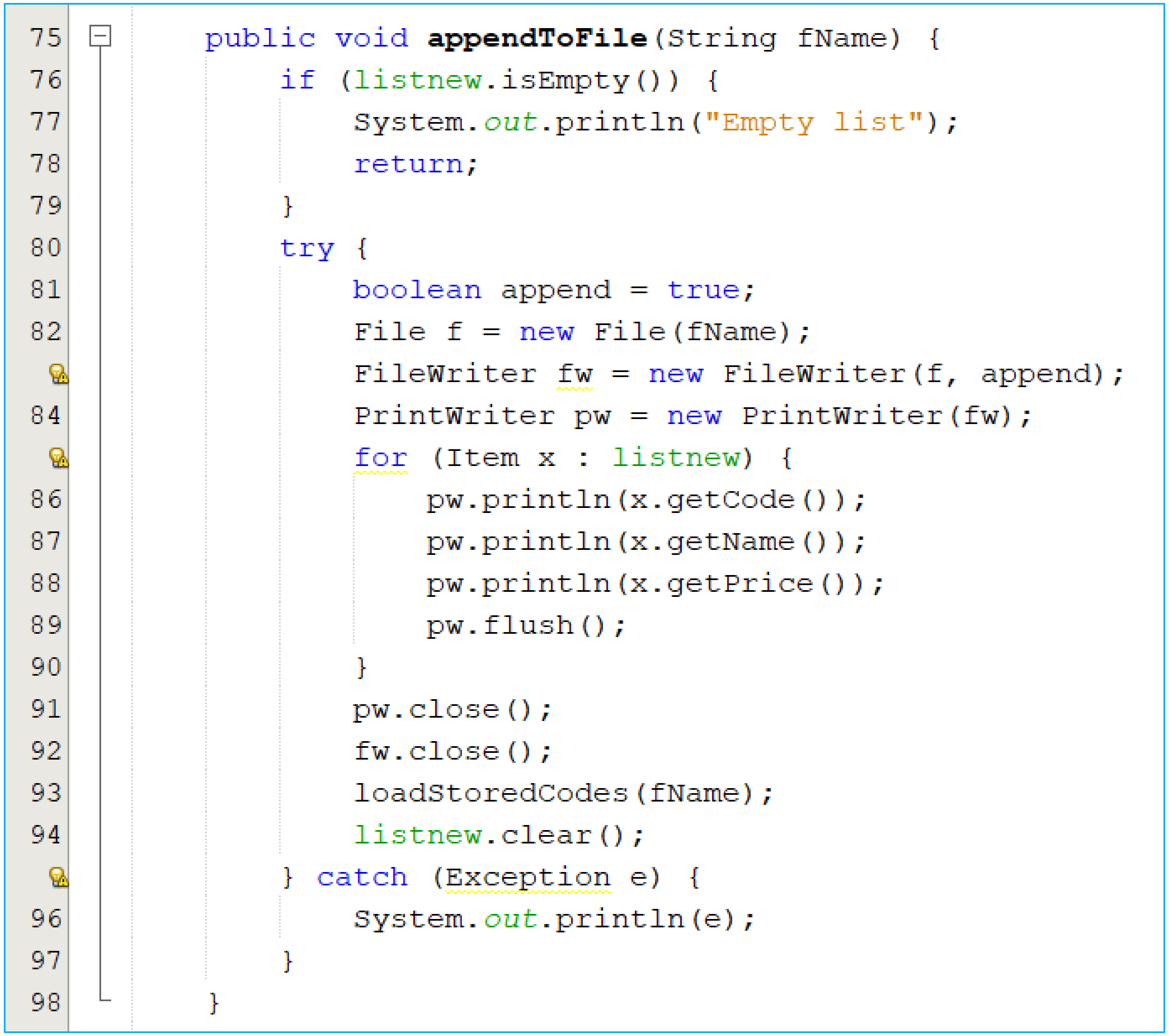
Access Text Files …: Case study 2- Implementations



```java
    private boolean valid(String aCode) {
        int i;
        for (i = 0; i < storedCodes.size(); i++) {
            if (aCode.equals(storedCodes.get(i))) {
                return false;
            }
        }
        for (i = 0; i < listnew.size(); i++) {
            if (aCode.equals(listnew.get(i).getCode())) {
                return false;
            }
        }
        return true;
    }

    private int find(String aCode) {
        for (int i = 0; i < listnew.size(); i++) {
            if (listnew.get(i).getCode().equals(aCode)) {
                return i;
            }
        }
        return -1;
    }
```

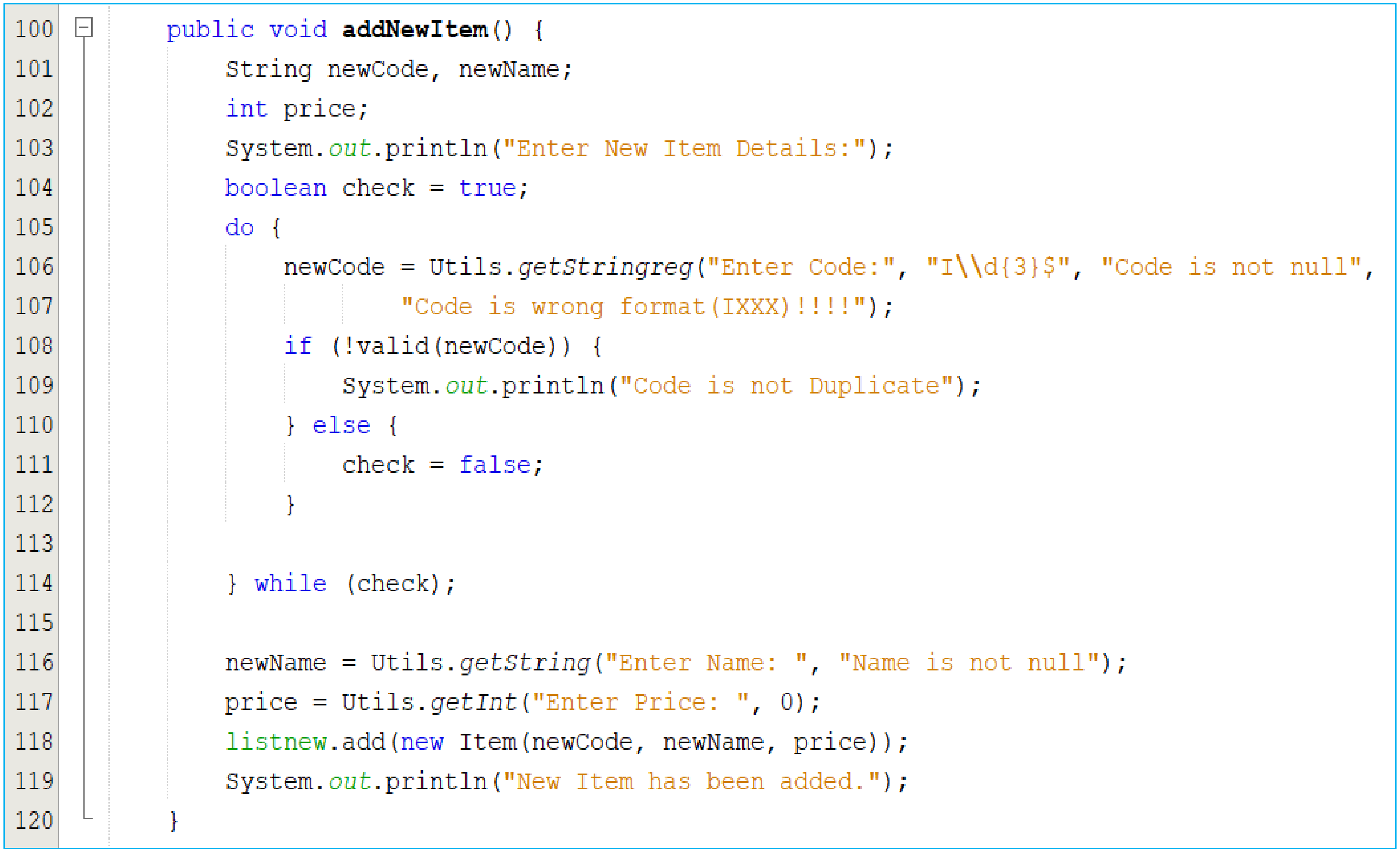
TRƯỜNG ĐẠI HỌC FPT

Access Text Files …: Case study 2- Implementations



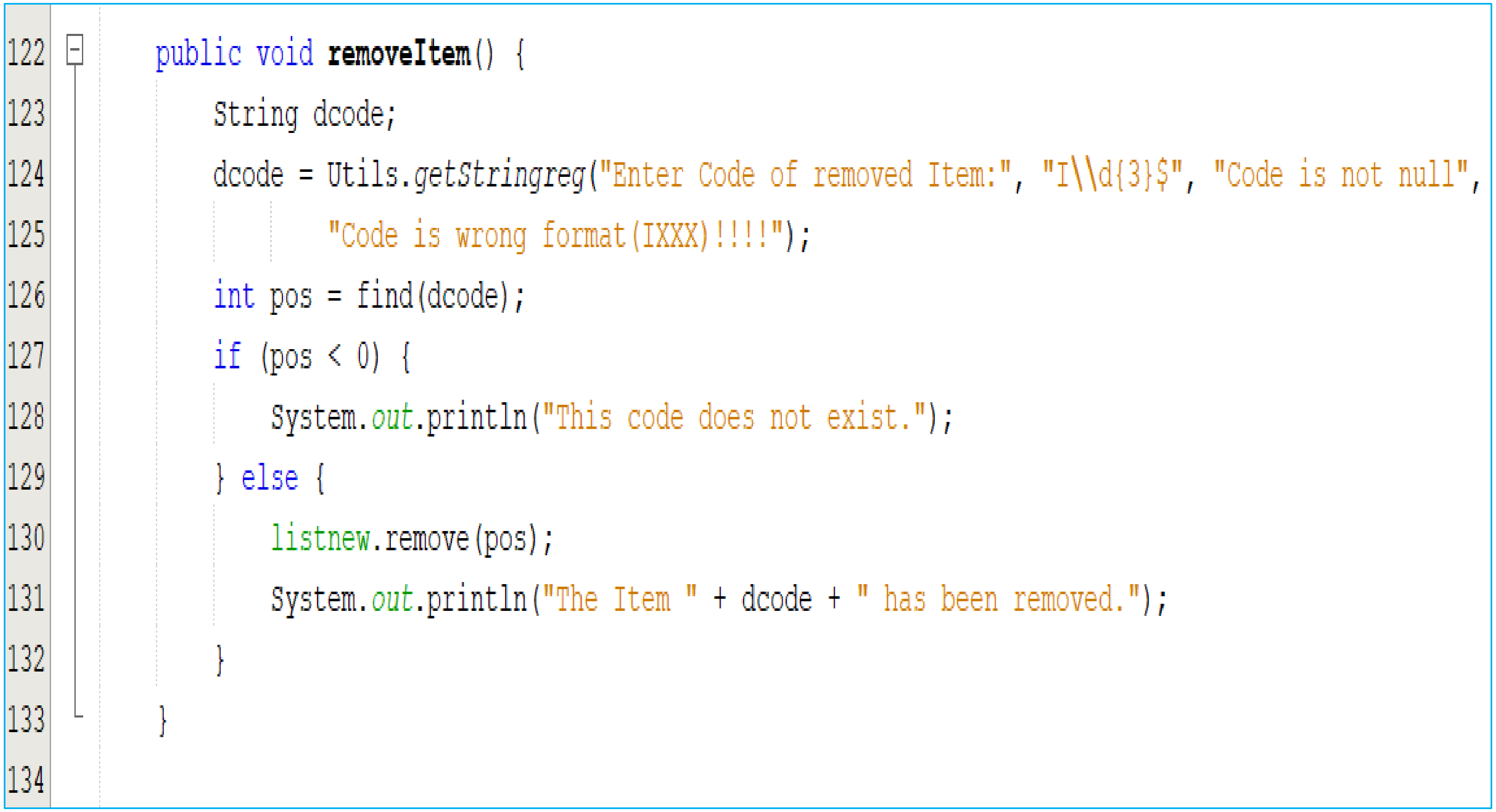
```java
    public void appendToFile(String fName) {
        if (listnew.isEmpty()) {
            System.out.println("Empty list");
            return;
        }
        try {
            boolean append = true;
            File f = new File(fName);
            FileWriter fw = new FileWriter(f, append);
            PrintWriter pw = new PrintWriter(fw);
            for (Item x : listnew) {
                pw.println(x.getCode());
                pw.println(x.getName());
                pw.println(x.getPrice());
                pw.flush();
            }
            pw.close();
            fw.close();
            loadStoredCodes(fName);
            listnew.clear();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
```

Access Text Files …: Case study 2- Implementations



```java
    public void addNewItem() {
        String newCode, newName;
        int price;
        System.out.println("Enter New Item Details:");
        boolean check = true;
        do {
            newCode = Utils.getStringreg("Enter Code:", "I\\d{3}$", "Code is not null",
                    "Code is wrong format(IXXX)!!!!");
            if (!valid(newCode)) {
                System.out.println("Code is not Duplicate");
            } else {
                check = false;
            }

        } while (check);

        newName = Utils.getString("Enter Name: ", "Name is not null");
        price = Utils.getInt("Enter Price: ", 0);
        listnew.add(new Item(newCode, newName, price));
        System.out.println("New Item has been added.");
    }
```

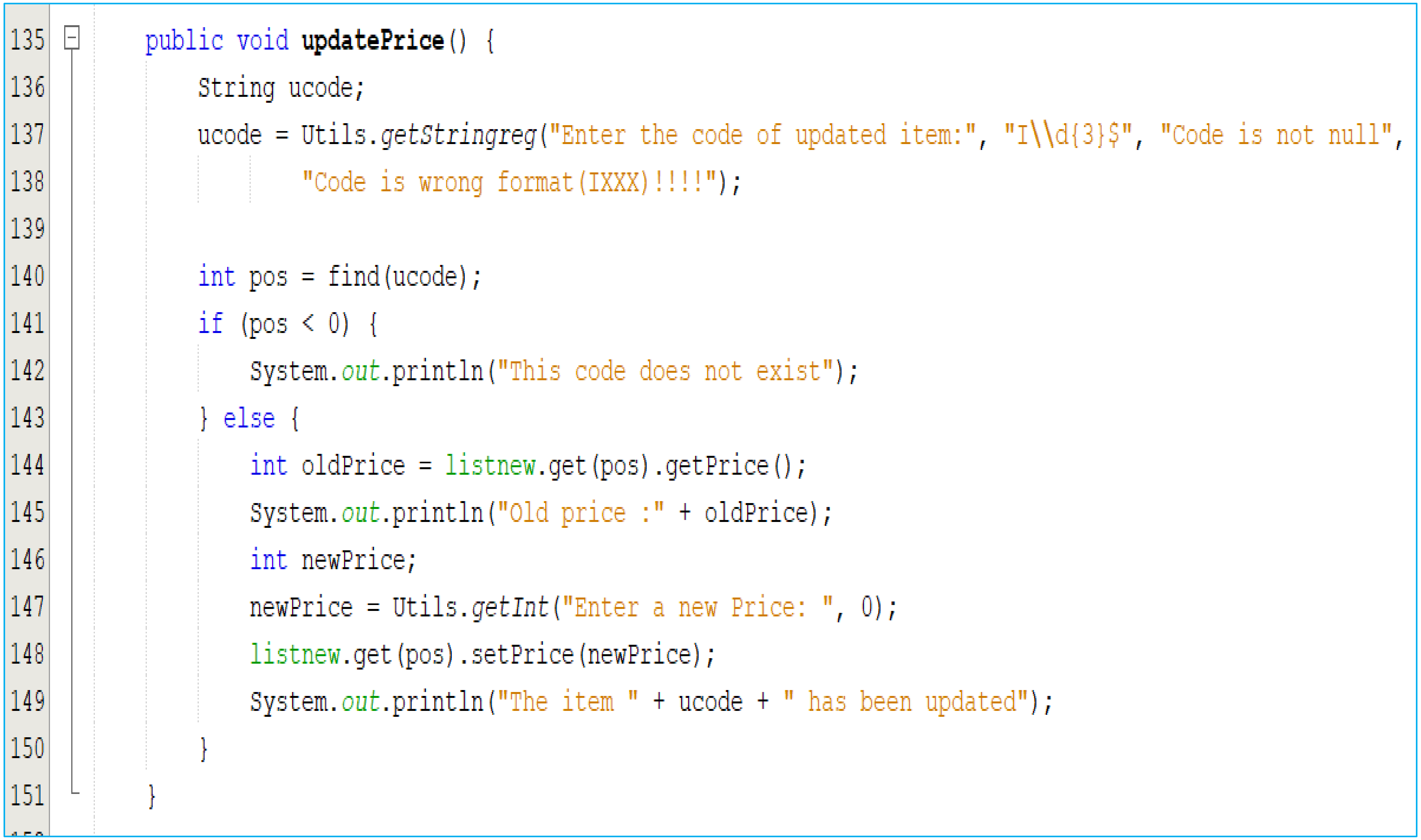
Access Text Files …: Case study 2- Implementations



```
public void removeItem() {
    String dcode;
    dcode = Utils.getStringreg("Enter Code of removed Item:", "I\\d{3}$", "Code is not null",
            "Code is wrong format(IXXX)!!!!");
    int pos = find(dcode);
    if (pos < 0) {
        System.out.println("This code does not exist.");
    } else {
        listnew.remove(pos);
        System.out.println("The Item " + dcode + " has been removed.");
    }
}
```

TRƯỜNG ĐẠI HỌC FPT

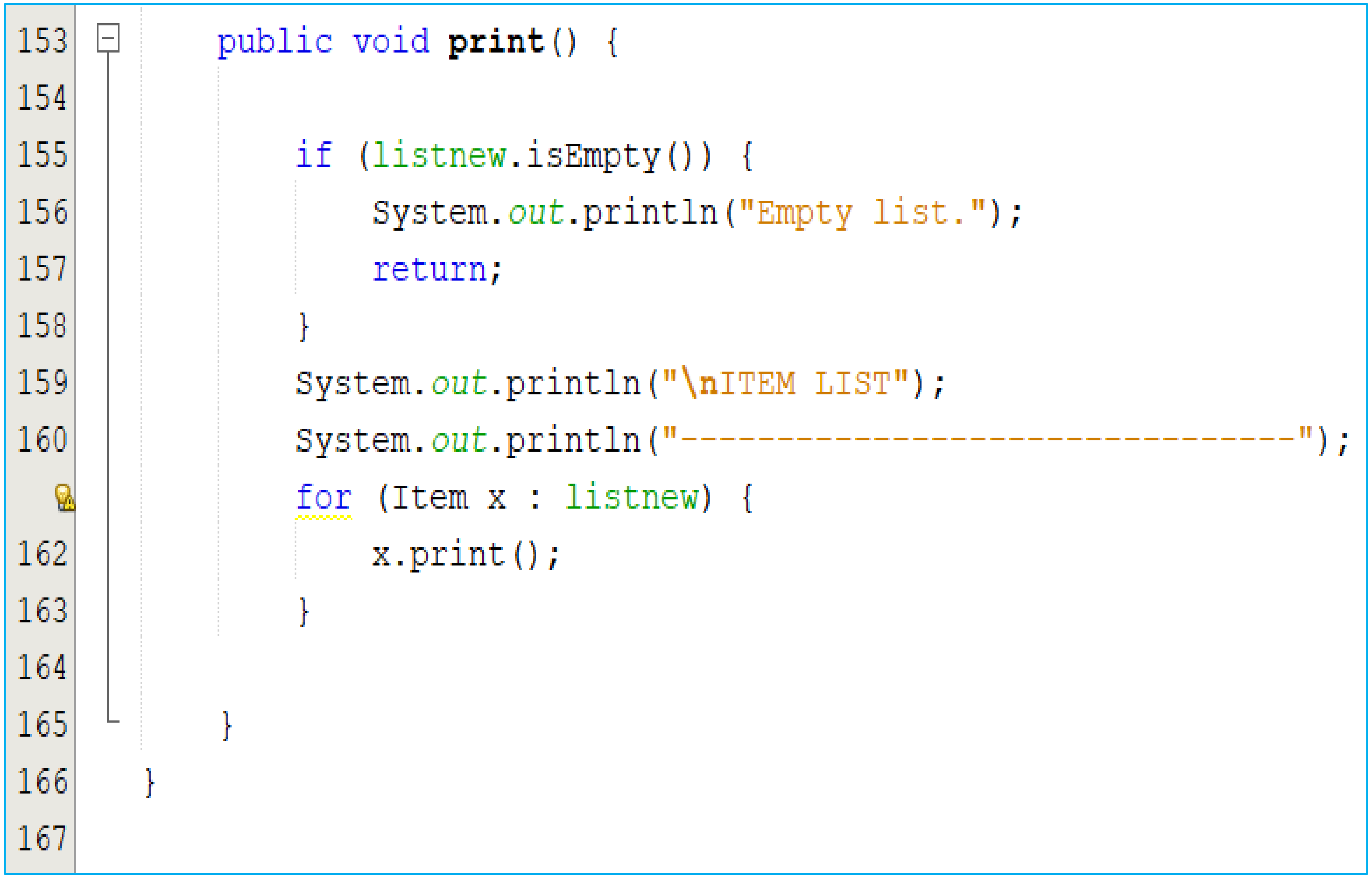
Access Text Files …: Case study 2- Implementations



```java
    public void updatePrice() {
        String ucode;
        ucode = Utils.getStringreg("Enter the code of updated item:", "I\\d{3}$", "Code is not null",
                "Code is wrong format(IXXX)!!!!");

        int pos = find(ucode);
        if (pos < 0) {
            System.out.println("This code does not exist");
        } else {
            int oldPrice = listnew.get(pos).getPrice();
            System.out.println("Old price :" + oldPrice);
            int newPrice;
            newPrice = Utils.getInt("Enter a new Price: ", 0);
            listnew.get(pos).setPrice(newPrice);
            System.out.println("The item " + ucode + " has been updated");
        }
    }
```

TRƯỜNG ĐẠI HỌC FPT

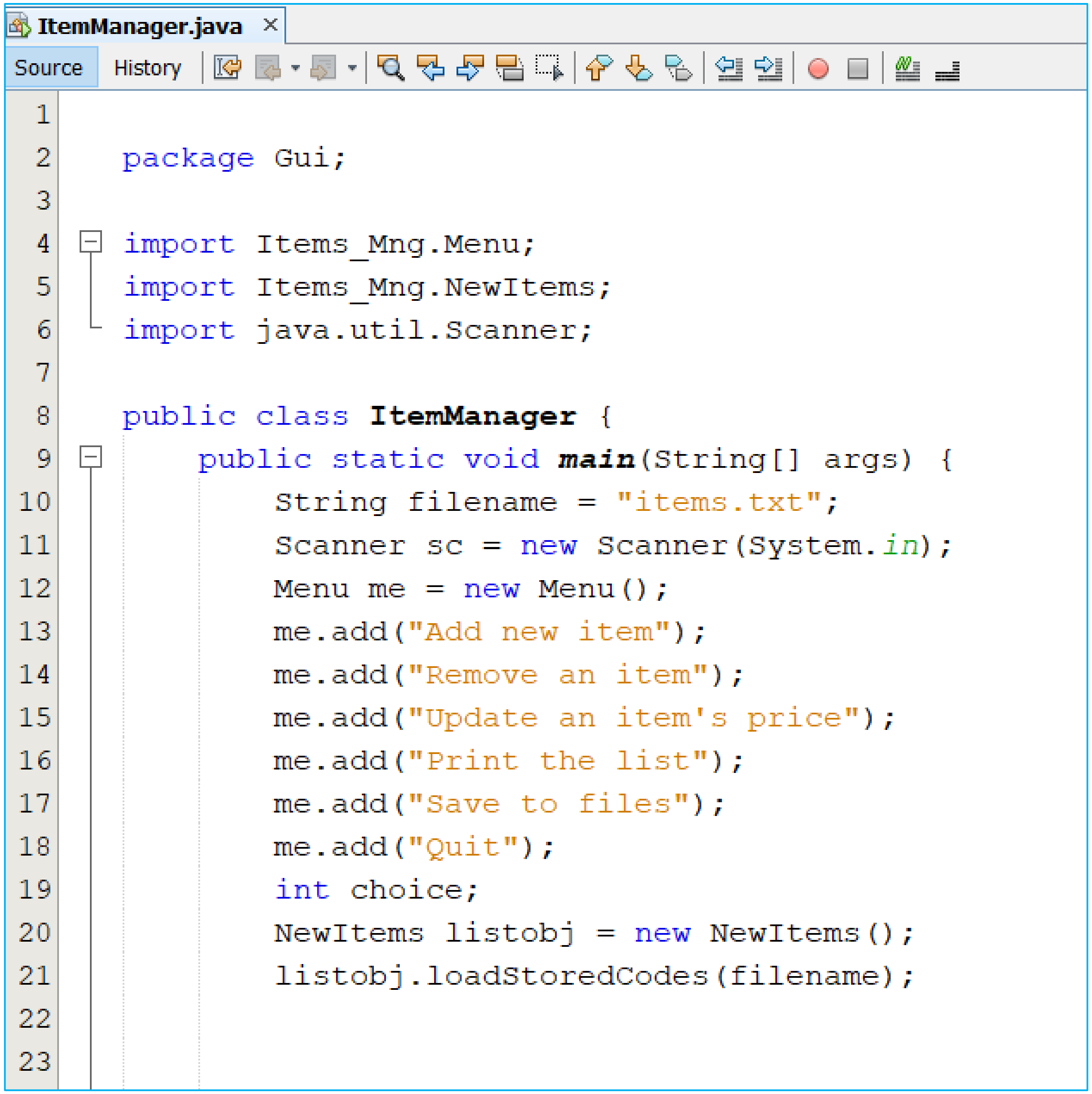
Access Text Files …: Case study 2- Implementations



```
    public void print() {

        if (listnew.isEmpty()) {
            System.out.println("Empty list.");
            return;
        }
        System.out.println("\nITEM LIST");
        System.out.println("-------------------------------");
        for (Item x : listnew) {
            x.print();
        }

    }
}

```

TRƯỜNG ĐẠI HỌC FPT

Access Text Files …: Case study 2- Implementations

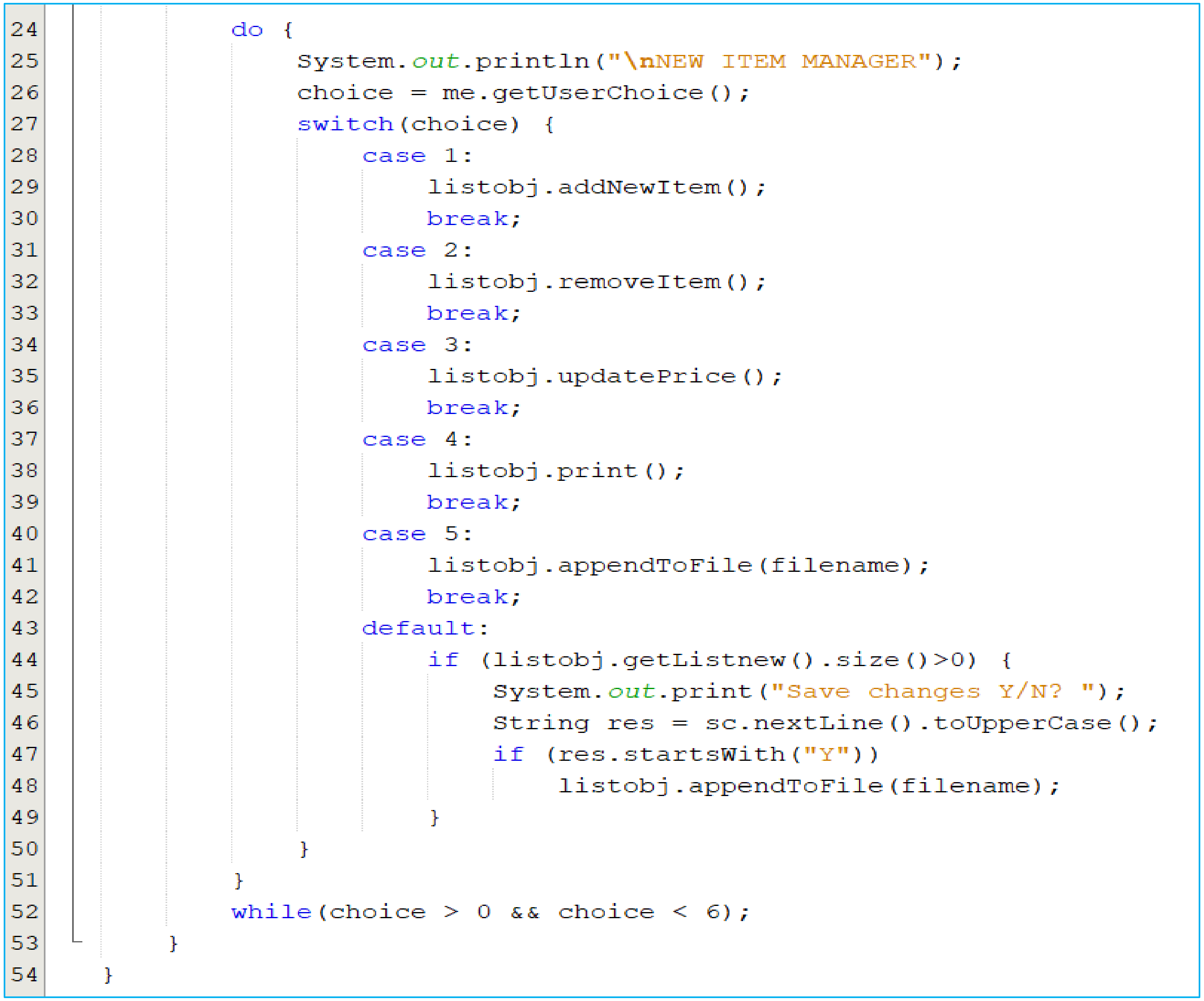


ItemManager.java ×

Source History

```java

package Gui;

import Items_Mng.Menu;
import Items_Mng.NewItems;
import java.util.Scanner;

public class ItemManager {
    public static void main(String[] args) {
        String filename = "items.txt";
        Scanner sc = new Scanner(System.in);
        Menu me = new Menu();
        me.add("Add new item");
        me.add("Remove an item");
        me.add("Update an item's price");
        me.add("Print the list");
        me.add("Save to files");
        me.add("Quit");
        int choice;
        NewItems listobj = new NewItems();
        listobj.loadStoredCodes(filename);


```

Access Text Files …: Case study 2- Implementations



```
        do {
            System.out.println("\nNEW ITEM MANAGER");
            choice = me.getUserChoice();
            switch(choice) {
                case 1:
                    listobj.addNewItem();
                    break;
                case 2:
                    listobj.removeItem();
                    break;
                case 3:
                    listobj.updatePrice();
                    break;
                case 4:
                    listobj.print();
                    break;
                case 5:
                    listobj.appendToFile(filename);
                    break;
                default:
                    if (listobj.getListnew().size()>0) {
                        System.out.print("Save changes Y/N? ");
                        String res = sc.nextLine().toUpperCase();
                        if (res.startsWith("Y"))
                            listobj.appendToFile(filename);
                    }
            }
        }
        while(choice > 0 && choice < 6);
    }
}
```

TRƯỜNG ĐẠI HỌC FPT

Access Text Files …: Read UTF-8 File content

UTF8 content is stored in compressed format→ a character will be stored in 1 to 3 bytes.
Before reading UTF, decompressing is needed.

```
String content="";
FileInputStream f = new FileInputStream(filename);
InputStreamReader isr = new InputStreamReader(f, "UTF8");
int ch;
while ((ch = in.read()) > -1) content+=(char)ch;
```

For read bytes

For read a unicode character

Or "UTF-8"

```
String content="", s;
FileInputStream f = new FileInputStream(filename);
InputStreamReader isr = new InputStreamReader(f, "UTF8");
BufferedReader br = new BufferedReader (isr);
while ( (s= br.readline())!=null) content += s + "\n";
```

For read a unicode character or string.

5- Access binary files

- Binary streams.
  - Low-level streams: reading/writing data byte-by-byte.
  - High-level stream: reading/writing general-format data (primitives – group of bytes that store typed-values)

Access binary files… The java.io.RandomAccessFile class

- It is used to read or modify data in a file that is compatible with the stream, or reader, or writer model
- It supports:
  - Get the file pointer
  - Get the length of the file
  - Seeking to any position within a file
  - Reading & writing single byte/groups of bytes, treated as higher-level data types
  - Close file.

Access binary files … java.io.RandomAccessFile class…

- Constructors

  RandomAccessFile(String *file*, String *mode*)

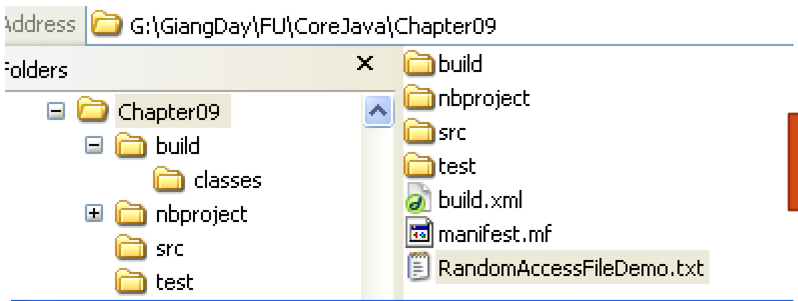
  RandomAccessFile(File *file*, String *mode*)

  - Mode “r” to open the file for reading only
  - Mode “rw” to open for both reading and writing
  - Mode “rws” is same as rw and any changes to the file’s content or metadata (file attributes) take place **immediately**
  - Mode “rwd” is same as rw, and changes to the file content, but **not** its **metadata**, take place immediately. Its metadata are upadated only when the file is closed.

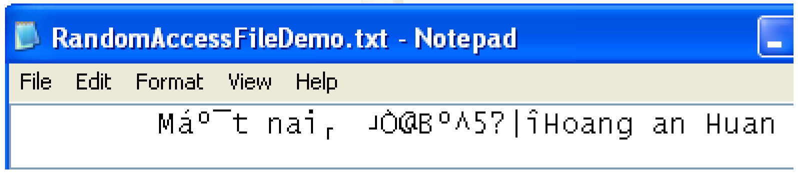
Access binary files … java.io.RandomAccessFile class…

A demo. for write data to a file then read data from the file

The try…catch statement must be used when accessing file – checked exception



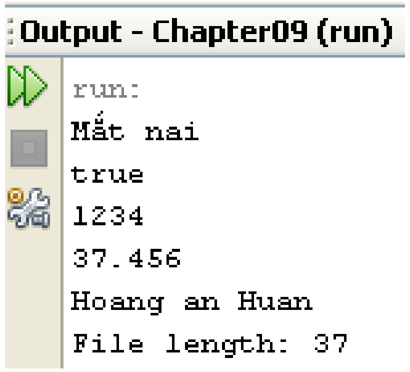
```
Address  G:\GiangDay\FU\CoreJava\Chapter09
Folders                         build
  ⊟ Chapter09                   nbproject
     ⊟ build                    src
          classes               test
     ⊞ nbproject                build.xml
       src                      manifest.mf
       test                     RandomAccessFileDemo.txt
```



RandomAccessFileDemo.txt - Notepad

File Edit Format View Help

```
      Má°¯t nai╷  ┘Ò@B°^5?|îHoang an Huan
```



Output - Chapter09 (run)

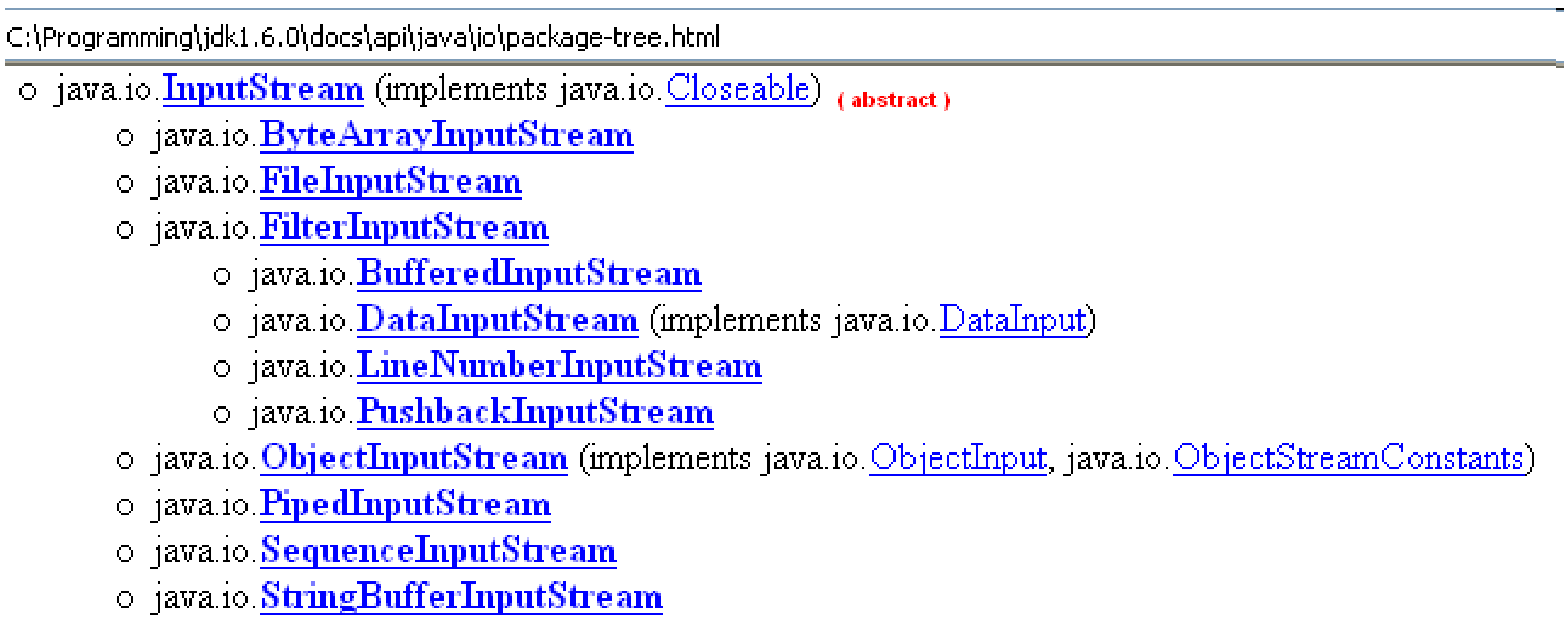
```
run:
Mắt nai
true
1234
37.456
Hoang an Huan
File length: 37
```

WRITE

READ

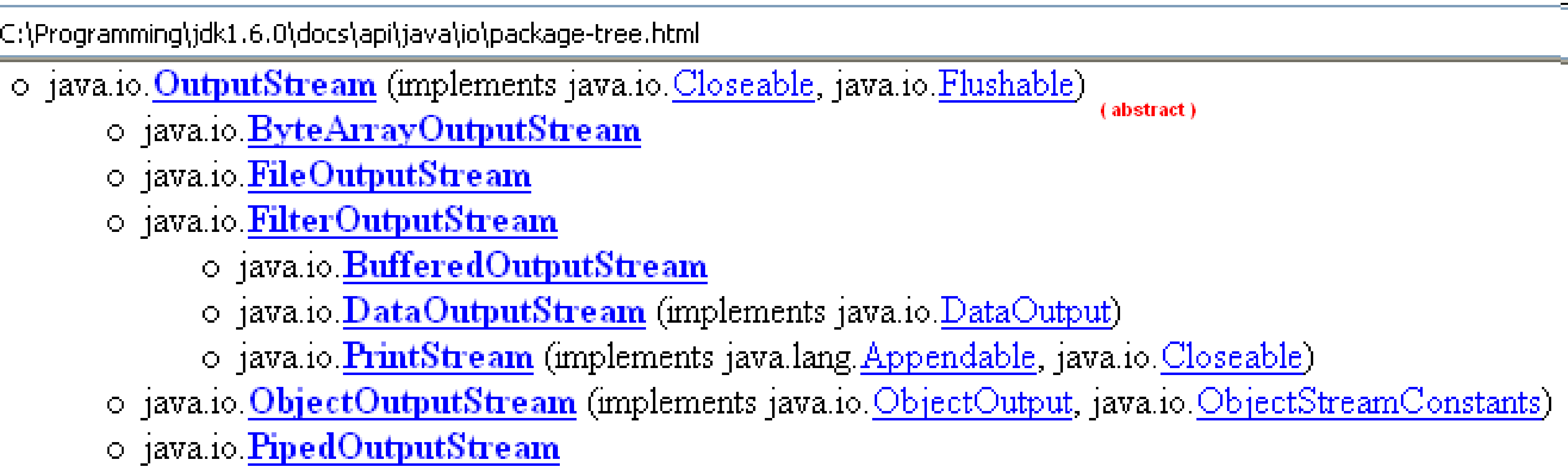
```java
/* Use the RandomAccessFile class to write/read some data */
import java.io.*;
public class RandomAccessFileDemo {
   public static void main (String[] args){
       String fName="RandomAccessFileDemo.txt";
       String S1= "Mắt nai"; boolean b=true; int n= 1234;
       double x= 37.456; String S2="Hoang an Huan";
       byte[] ar= new byte[100]; // for reading ASCII characters
       try {
         RandomAccessFile f= new RandomAccessFile(fName, "rw");
         // Write data , positions: 0,1,2,3,4
         f.writeUTF(S1); f.writeBoolean(b);  f.writeInt(n);
         f.writeDouble(x); f.writeBytes(S2);
         // Read data
         f.seek(0);  // seek to BOF
         System.out.println(f.readUTF());
         System.out.println(f.readBoolean());
         System.out.println(f.readInt());
         System.out.println(f.readDouble());
         f.read(ar);
         System.out.println(new String (ar));
         System.out.println("File length: " + f.length());
         f.close();
       }
       catch (Exception e){
           System.out.println(e);
       }
   }
}
```

Access binary files… Binary Streams



C:\Programming\jdk1.6.0\docs\api\java\io\package-tree.html

- java.io.InputStream (implements java.io.Closeable) (abstract)
  - java.io.ByteArrayInputStream
  - java.io.FileInputStream
  - java.io.FilterInputStream
    - java.io.BufferedInputStream
    - java.io.DataInputStream (implements java.io.DataInput)
    - java.io.LineNumberInputStream
    - java.io.PushbackInputStream
  - java.io.ObjectInputStream (implements java.io.ObjectInput, java.io.ObjectStreamConstants)
  - java.io.PipedInputStream
  - java.io.SequenceInputStream
  - java.io.StringBufferInputStream



C:\Programming\jdk1.6.0\docs\api\java\io\package-tree.html

- java.io.OutputStream (implements java.io.Closeable, java.io.Flushable) (abstract)
  - java.io.ByteArrayOutputStream
  - java.io.FileOutputStream
  - java.io.FilterOutputStream
    - java.io.BufferedOutputStream
    - java.io.DataOutputStream (implements java.io.DataOutput)
    - java.io.PrintStream (implements java.lang.Appendable, java.io.Closeable)
  - java.io.ObjectOutputStream (implements java.io.ObjectOutput, java.io.ObjectStreamConstants)
  - java.io.PipedOutputStream

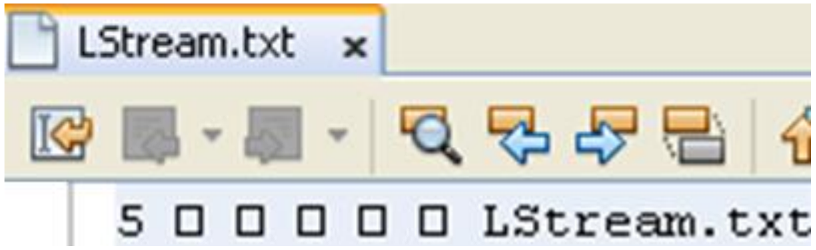
TRƯỜNG ĐẠI HỌC FPT

Access binary files…
Low-Level Binary Stream Demo.1

```
public class LowLevelStreamDemo {
    /**...*/
    public static void main(String[] args) {
        final char BLANK=32;
        final String fileName="LStream.txt";
        int[] a ={1, 2, 3, 4, 5};
        char n = '5';
        try {
            FileOutputStream os = new FileOutputStream(fileName);
            os.write(n);//begin writing
            os.write(BLANK);
            for(int i=0; i<5; i++){
                os.write(a[i]);
                os.write(BLANK);
            }
            for(int i=0; i<fileName.length(); i++){
                os.write(fileName.charAt(i));
            }
            os.close();
```

These values can not be greater than 127 because only the lower bytes are written to the file.

Write data to file



```
5 □ □ □ □ □ LStream.txt
```

We can not read these number in the file because of binary file. However, we can see characters.

Access binary files…
Low-Level Binary Stream Demo.1…

Read data from the file then print them out.

```
FileInputStream is = new FileInputStream(fileName);
int count = is.available();
System.out.println("The size of file is " + count + " bytes");
System.out.println("The content of file: ");
//read first char
byte[] bytes = new byte[1];
is.read(bytes);
System.out.print(new String(bytes));
//read blank
is.read(bytes);
System.out.print(new String(bytes));
//read int number
for(int i=0; i<5; i++){
    int tmp = is.read();
    is.read(bytes);
    System.out.print(tmp + new String(bytes));
}
bytes = new byte[11];
is.read(bytes);
System.out.println(new String(bytes));
is.close();
}catch(IOException e){
    e.printStackTrace();
}
}
}
```

Read a byte: '5'

Convert array of characters to string for printing them easier.

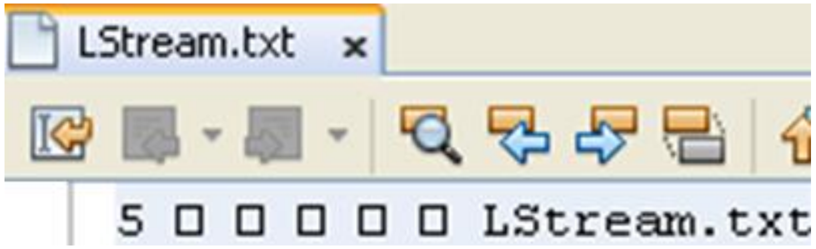
Read the blank

Read the blank

Read a number

Read filename stored at the end of the file

```
The size of file is 23 bytes
The content of file:
5 1 2 3 4 5 LStream.txt
```



```
5 □ □ □ □ □ LStream.txt
```

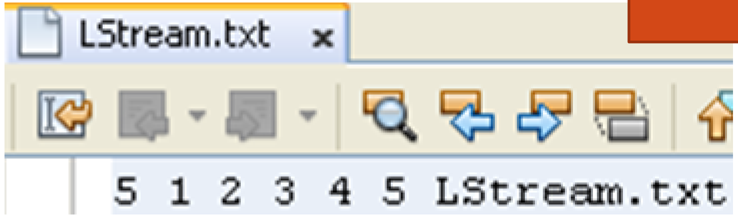
Access binary files…
Low-Level Binary Stream Demo.2

```
public class LowLevelStreamDemo {
    /**...*/
    public static void main(String[] args) {
        final char BLANK=32;
        final String fileName="LStream.txt";
        int[] a ={1, 2, 3, 4, 5};
        char n = '5';
        try {
            FileOutputStream os = new FileOutputStream(fileName);
            os.write(n);//begin writing
            os.write(BLANK);
            for(int i=0; i<5; i++){
                os.write(Character.forDigit(a[i],10));
                os.write(BLANK);
            }
            for(int i=0; i<fileName.length(); i++){
                os.write(fileName.charAt(i));
            }
            os.close();
```

This demo. Is the same as the previous one. But, all small number will be converted to digits then write them to the file

Write data to file

Now, we can see all the file content because they are characters


LStream.txt

```
5 1 2 3 4 5 LStream.txt
```

TRƯỜNG ĐẠI HỌC FPT

Access binary files…
Low-Level Binary Stream Demo.2…

Read data from the file

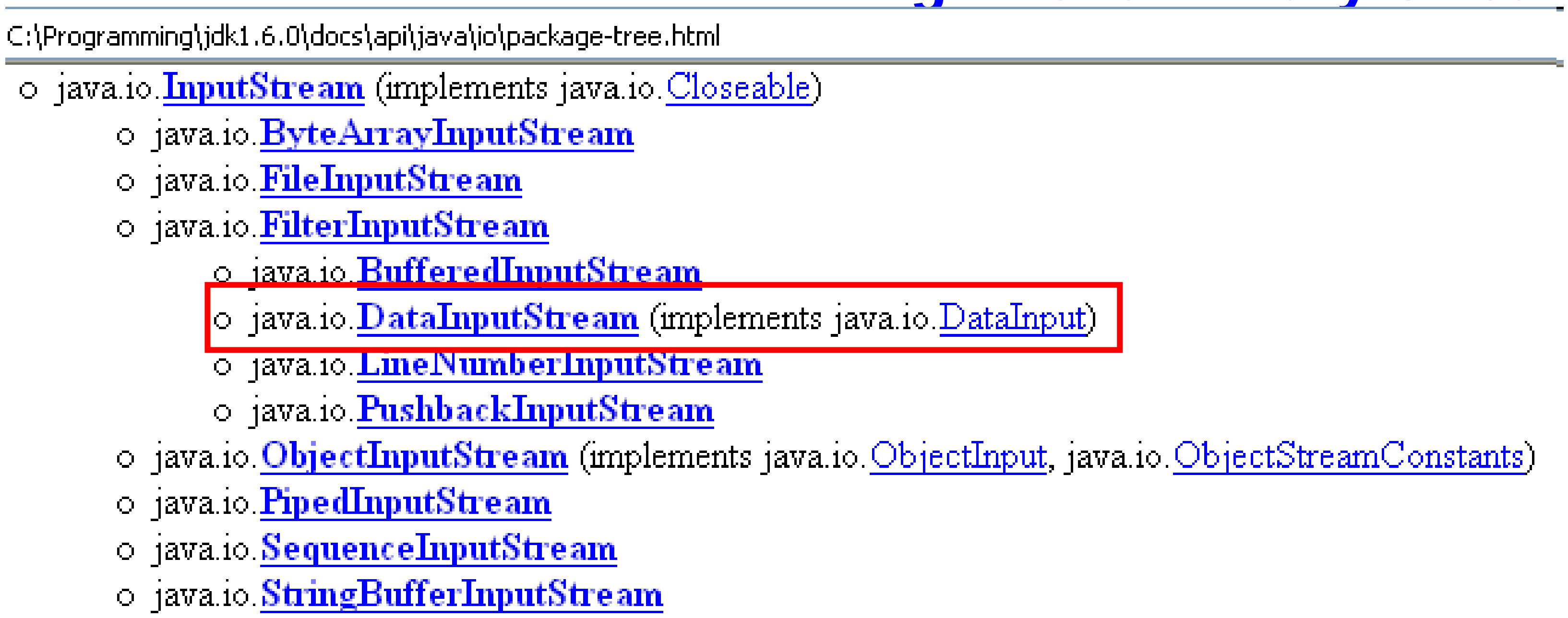
```
            FileInputStream is = new FileInputStream(fileName);
            int count = is.available();
            System.out.println("The size of file is " + count + " bytes");
            byte[] bytes = new byte[count];
            int readCount = is.read(bytes);
            System.out.println("The content of file: ");
            System.out.println(new String(bytes));
            System.out.println("Number of read bytes: " + readCount);
            is.close();
        }catch(IOException e){
            e.printStackTrace();
        }
}
```

```
The size of file is 23 bytes
The content of file:
5 1 2 3 4 5 LStream.txt
Number of read bytes: 23
```

Access binary files
High-Level Binary Stream

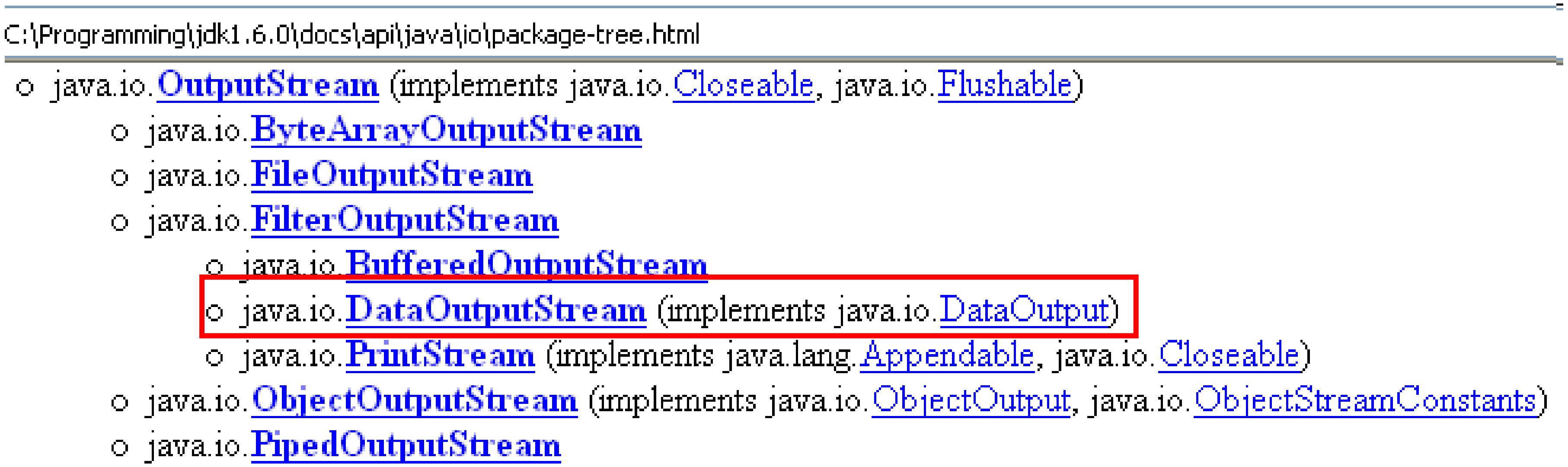
- More often than not bytes to be read or written constitute higher-level information (int, String, …)
- The most common of high-level streams extend from the super classes FilterInputStream and FilterOutputStream.
- Do not read/write from input/output devices such as files or sockets; rather, they read/write from other streams
  - DataInputStream/ DataOutputStream
    - Constructor argument: InputStream/ OutputStream
    - Common methods: readXXX, writeXXX
  - BufferedInputStream/ BufferedOutputStream: supports read/write in large blocks
  - ….

Access binary files… High-Level Binary Streams



C:\Programming\jdk1.6.0\docs\api\java\io\package-tree.html

- java.io.**InputStream** (implements java.io.Closeable)
  - java.io.**ByteArrayInputStream**
  - java.io.**FileInputStream**
  - java.io.**FilterInputStream**
    - java.io.**BufferedInputStream**
    - java.io.**DataInputStream** (implements java.io.DataInput)
    - java.io.**LineNumberInputStream**
    - java.io.**PushbackInputStream**
  - java.io.**ObjectInputStream** (implements java.io.ObjectInput, java.io.ObjectStreamConstants)
  - java.io.**PipedInputStream**
  - java.io.**SequenceInputStream**
  - java.io.**StringBufferInputStream**

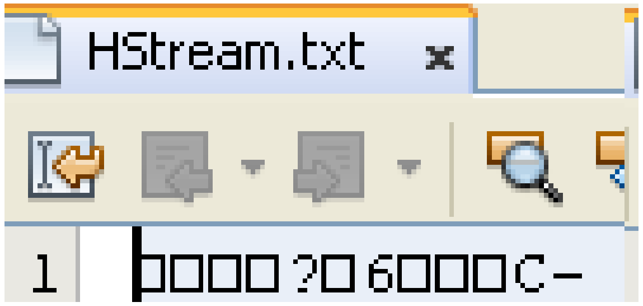


C:\Programming\jdk1.6.0\docs\api\java\io\package-tree.html

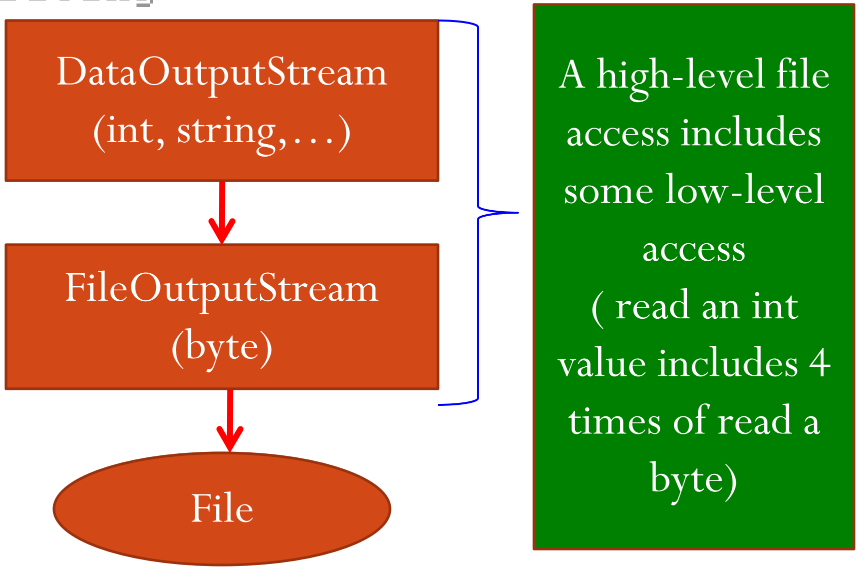
- java.io.**OutputStream** (implements java.io.Closeable, java.io.Flushable)
  - java.io.**ByteArrayOutputStream**
  - java.io.**FileOutputStream**
  - java.io.**FilterOutputStream**
    - java.io.**BufferedOutputStream**
    - java.io.**DataOutputStream** (implements java.io.DataOutput)
    - java.io.**PrintStream** (implements java.lang.Appendable, java.io.Closeable)
  - java.io.**ObjectOutputStream** (implements java.io.ObjectOutput, java.io.ObjectStreamConstants)
  - java.io.**PipedOutputStream**

Access binary files…
High-Level Binary Stream Demo.

```
public class HighLevelStreamDemo {
    /**...*/
    public static void main(String[] args) {
        final char BLANK=32;
        final String fileName="HStream.txt";
        int[] a ={1, 2, 3, 4, 5};
        char n = '5';
        try {
            FileOutputStream os = new FileOutputStream(fileName);
            DataOutputStream ds = new DataOutputStream(os);
            ds.writeChar(n);//begin writing
            ds.writeChar(BLANK);
            for(int i=0; i<5; i++){
                ds.writeInt(a[i]);
                ds.writeChar(BLANK);
            }
            ds.writeUTF(fileName);
            ds.close();
            os.close();
```



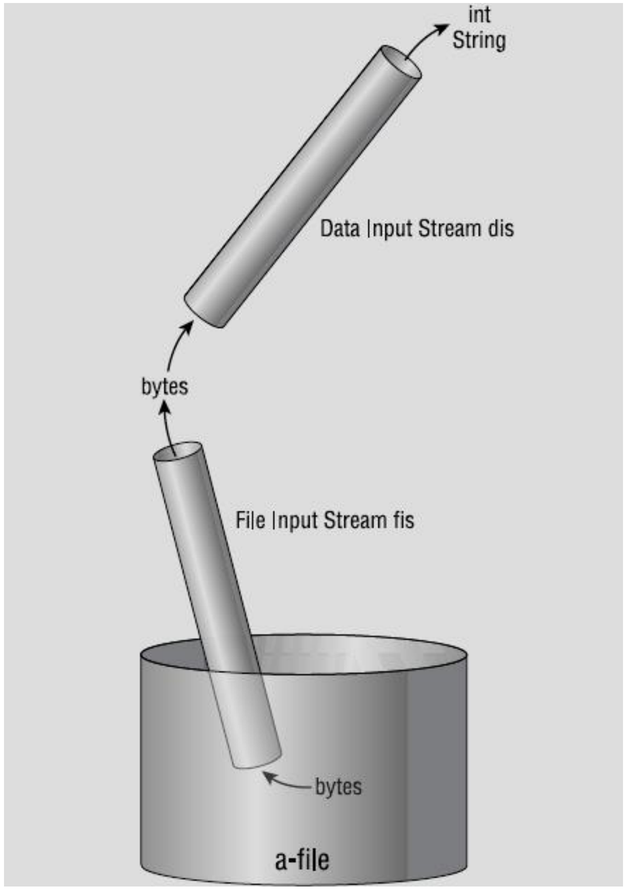
```
1    □□□□?□6□□□C-
```


DataOutputStream
(int, string,…)
FileOutputStream
(byte)
File
A high-level file access includes some low-level access
(read an int value includes 4 times of read a byte)

Access binary files…
High-Level Binary Stream Demo. …

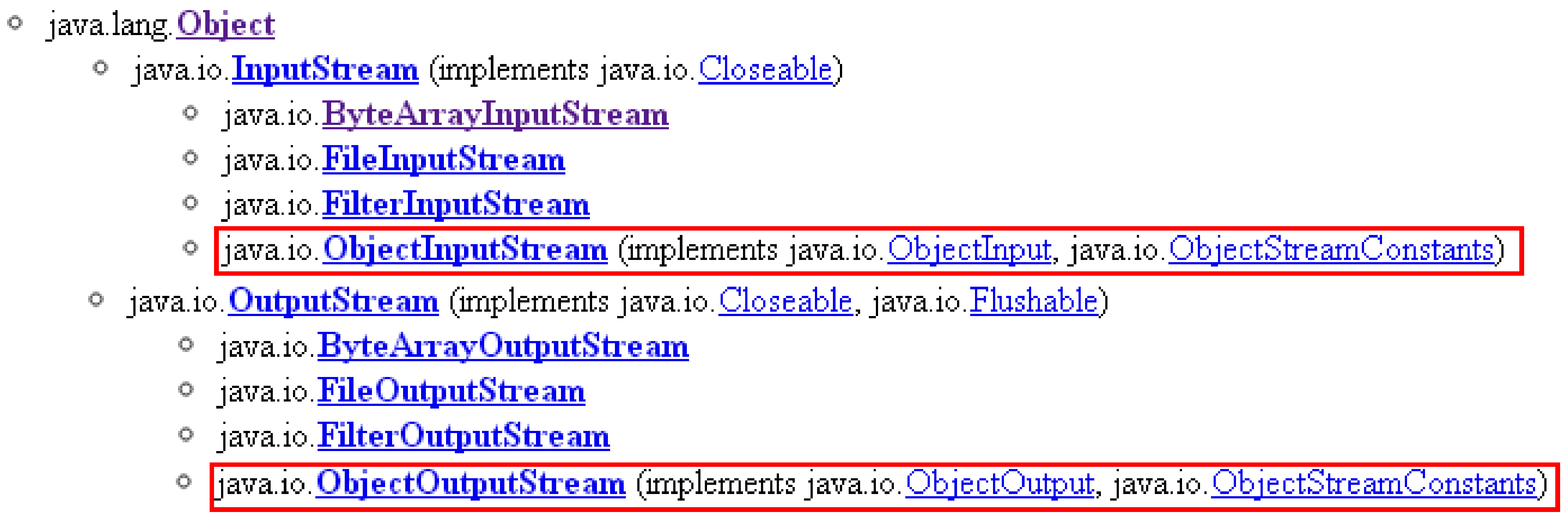
```
        FileInputStream is = new FileInputStream(fileName);
        DataInputStream dis = new DataInputStream(is);
        int count = dis.available();
        System.out.println("The size of file is " + count + " bytes");
        System.out.println("The content of file: ");
        System.out.print(dis.readChar());
        System.out.print(dis.readChar());
        for(int i=0; i<5; i++){
            System.out.print(dis.readInt());
            System.out.print(dis.readChar());
        }
        System.out.println(dis.readUTF());
        dis.close();
        is.close();
    }catch(IOException e){
        e.printStackTrace();
    }
  }
}
```

```
The size of file is 47 bytes
The content of file:
5 1 2 3 4 5 HStream.txt
```



6- Access Object Files

- 2 Object streams :Object Input stream, Object Output stream



- java.lang.Object
  - java.io.InputStream (implements java.io.Closeable)
    - java.io.ByteArrayInputStream
    - java.io.FileInputStream
    - java.io.FilterInputStream
    - java.io.ObjectInputStream (implements java.io.ObjectInput, java.io.ObjectStreamConstants)
  - java.io.OutputStream (implements java.io.Closeable, java.io.Flushable)
    - java.io.ByteArrayOutputStream
    - java.io.FileOutputStream
    - java.io.FilterOutputStream
    - java.io.ObjectOutputStream (implements java.io.ObjectOutput, java.io.ObjectStreamConstants)

Serialization is a task which will concate all data of an object to a byte stream then it can be written to a datasource. **Static and transient data can not be serialized.**
De-serialization is a task which will read a byte stream from a datasourse , split the stream to fields then assign them to data fields of an object appropriately.
Transient fields are omitted when an object is serialized.

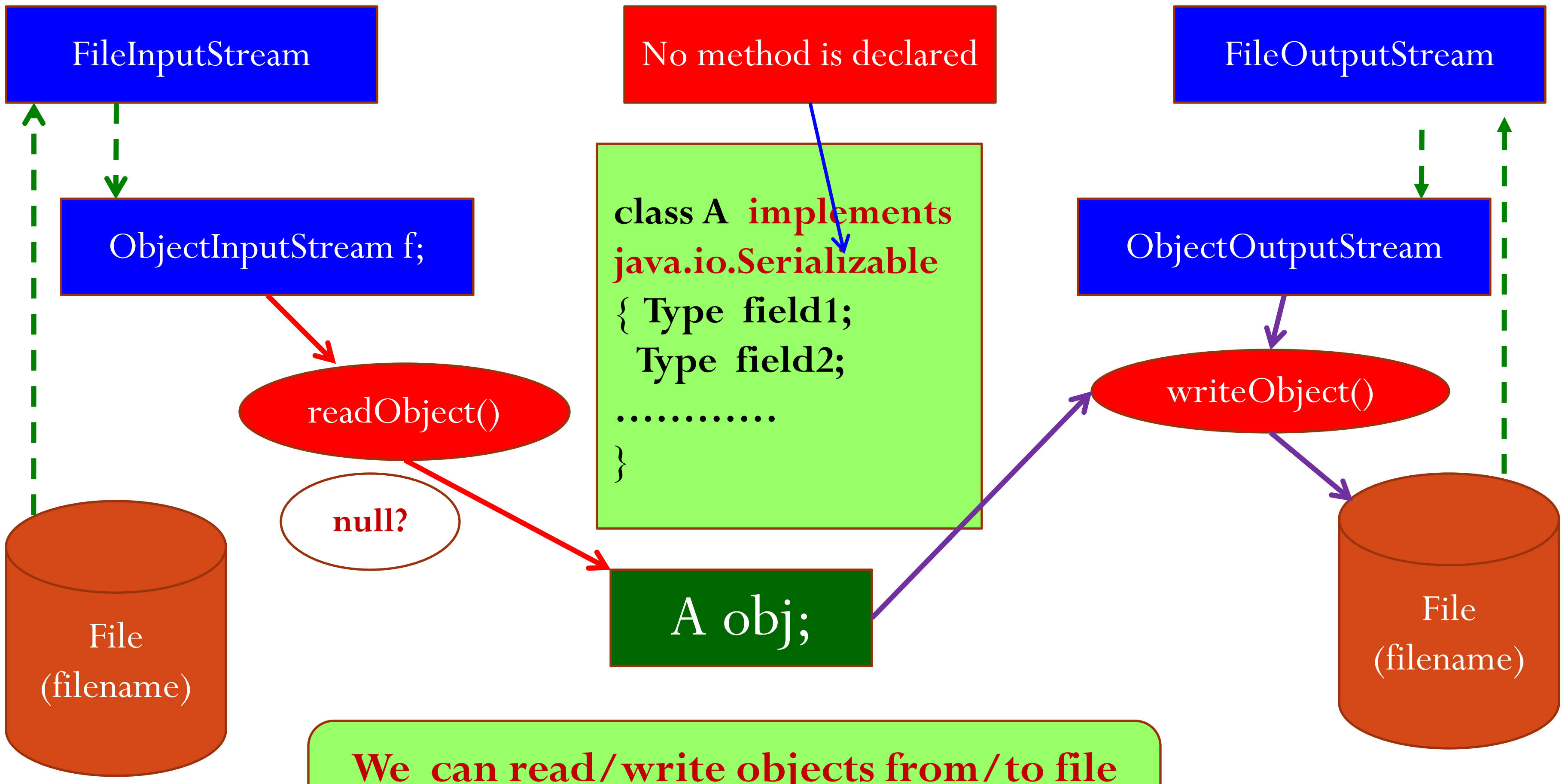
Serialization

- The process of writing an object is called *serialization.*
- Use java.io.ObjectOutputStream to serialize an object.
- It is only an object's data that is serialized, not its class definition.
- When an object output stream serializes an object that contains references to other object, every referenced object is serialized along with the original object.
- Not all data is written.
  - static fields are not
  - transient fields are also not serialized

De-serialization

- De-serialization is to convert a serialized representation into a replica of the original object.
- Use java.io.ObjectInputStream to deserialize an object.
- When an object is serialized, it will probably be deserialized by a different JVM.
- Any JVM that tries to deserialize an object must have access to that object’s class definition.

Access Object Files…: How to?



```
class A  implements
java.io.Serializable
{ Type  field1;
  Type  field2;
…………
}
```

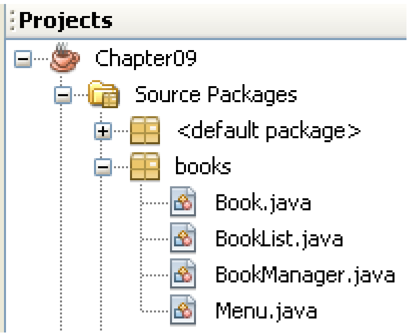
We can read/write objects from/to file using a program only.

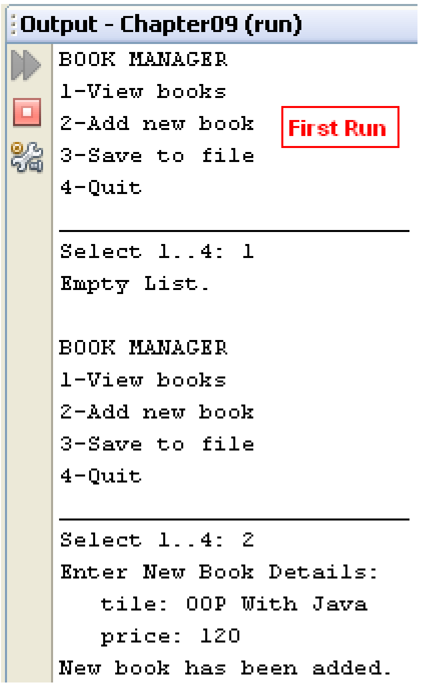
Access Object Files…: Case study 3 - Object Streams Demo.

Problem

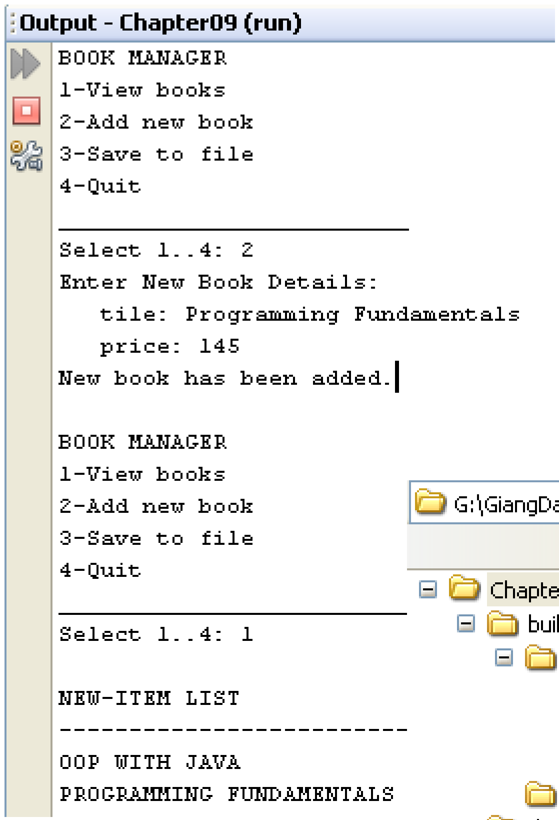
- Book <title, price>
- Write a Java program that allows user:
  - View books in the file books.dat
  - Append a book to the file
- Read/ Write books as binary objects from/to the file.

Access Object Files…: Case Study 3 - Design

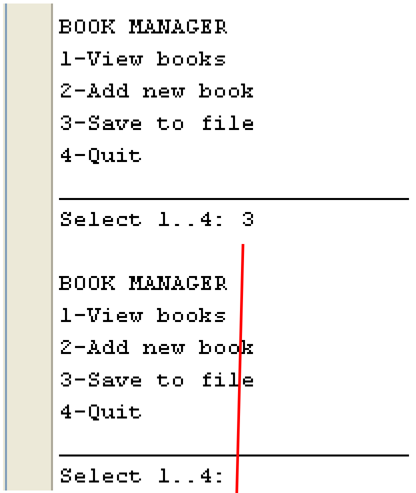




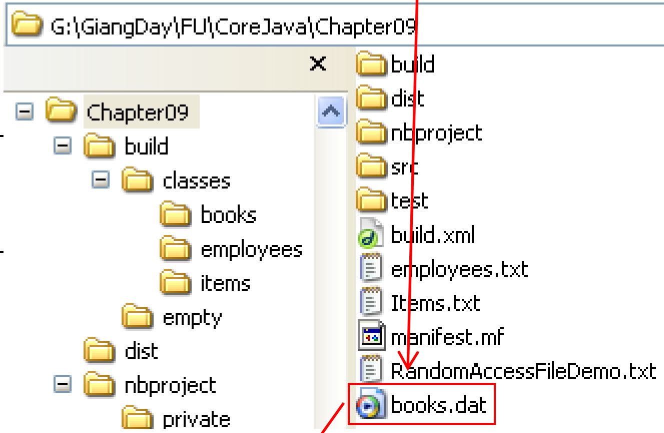
```
BOOK MANAGER
1-View books
2-Add new book
3-Save to file
4-Quit
_______________________________
Select 1..4: 1
Empty List.

BOOK MANAGER
1-View books
2-Add new book
3-Save to file
4-Quit
_______________________________
Select 1..4: 2
Enter New Book Details:
    tile: OOP With Java
    price: 120
New book has been added.
```

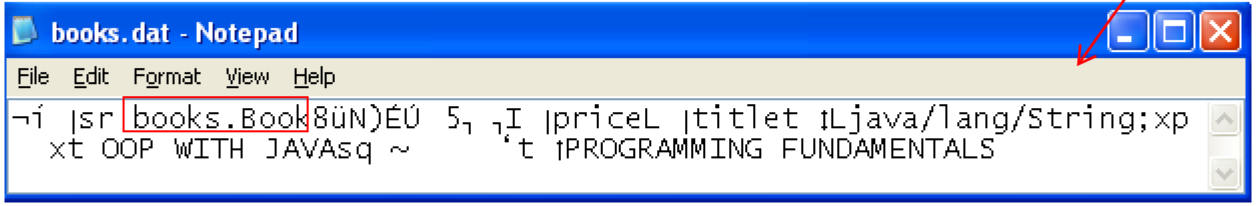


```
BOOK MANAGER
1-View books
2-Add new book
3-Save to file
4-Quit
_______________________________
Select 1..4: 2
Enter New Book Details:
    tile: Programming Fundamentals
    price: 145
New book has been added.

BOOK MANAGER
1-View books
2-Add new book
3-Save to file
4-Quit
_______________________________
Select 1..4: 1

NEW-ITEM LIST
--------------------------
OOP WITH JAVA
PROGRAMMING FUNDAMENTALS
```



```
BOOK MANAGER
1-View books
2-Add new book
3-Save to file
4-Quit
_______________________________
Select 1..4: 3

BOOK MANAGER
1-View books
2-Add new book
3-Save to file
4-Quit
_______________________________
Select 1..4:
```





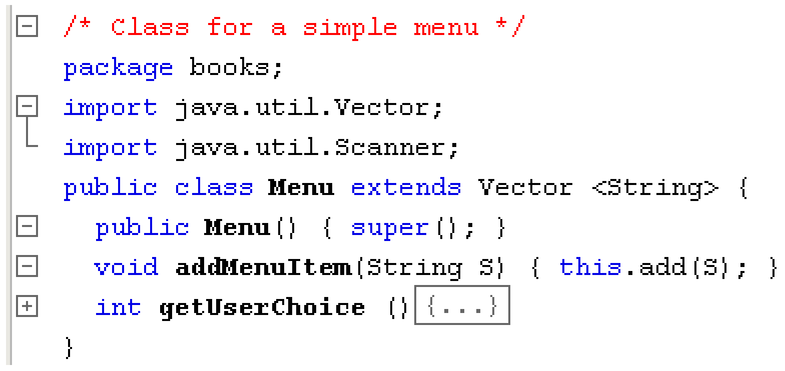
```
¬í ¦sr books.Book8ÜN)ÉÚ 5┐ ┐I ¦priceL ¦titlet ‡Ljava/lang/String;xp
 xt OOP WITH JAVAsq ~     ‘t ‡PROGRAMMING FUNDAMENTALS
```

Java serielize data of an object from the bottom of the declaration to the beginning.

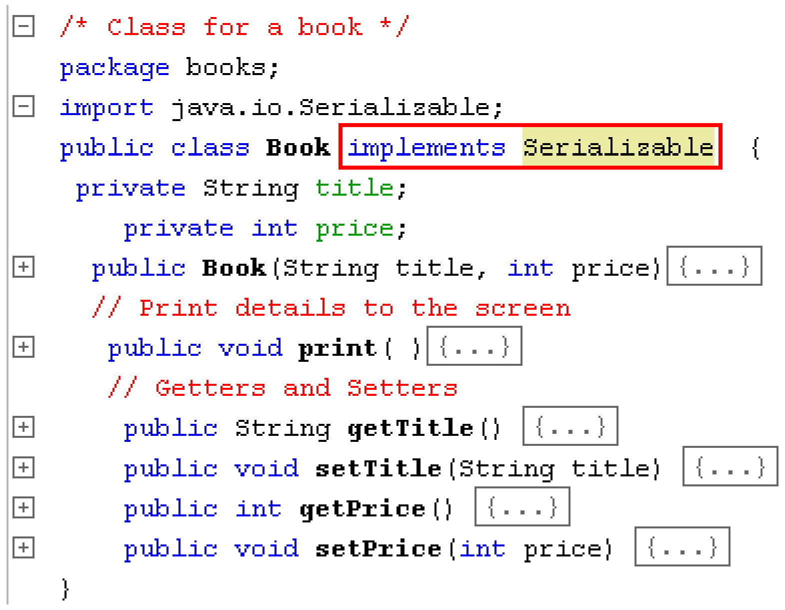
TRƯỜNG ĐẠI HỌC FPT

Access Object Files…: Case Study 3- Implementations

Refer to the case study 1, 2.
DO YOURSELF

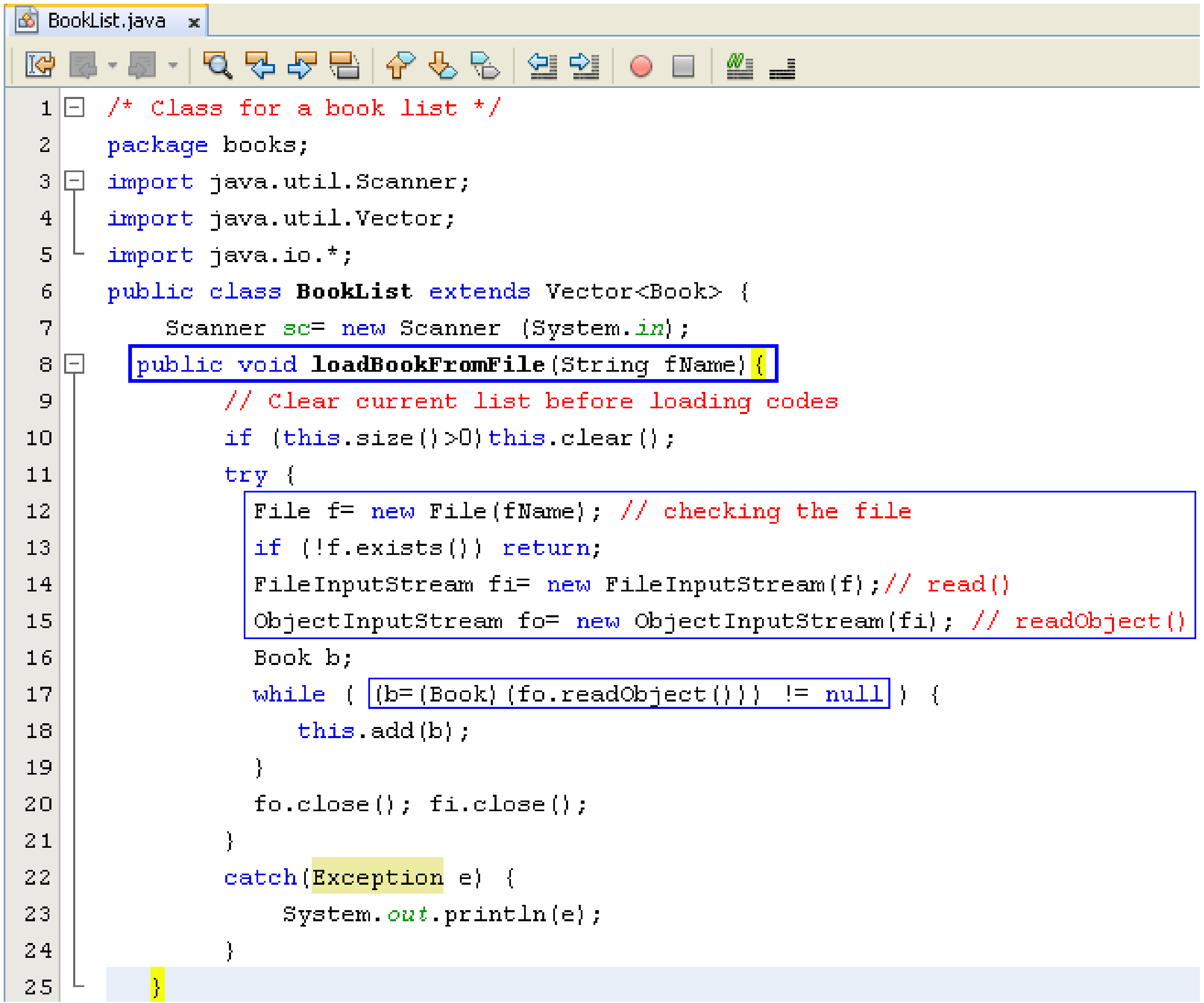


```
/* Class for a simple menu */
package books;
import java.util.Vector;
import java.util.Scanner;
public class Menu extends Vector <String> {
   public Menu() { super(); }
   void addMenuItem(String S) { this.add(S); }
   int getUserChoice () {...}
}
```



```
/* Class for a book */
package books;
import java.io.Serializable;
public class Book implements Serializable  {
 private String title;
    private int price;
  public Book(String title, int price){...}
  // Print details to the screen
   public void print( ){...}
   // Getters and Setters
    public String getTitle() {...}
    public void setTitle(String title) {...}
    public int getPrice() {...}
    public void setPrice(int price) {...}
}
```

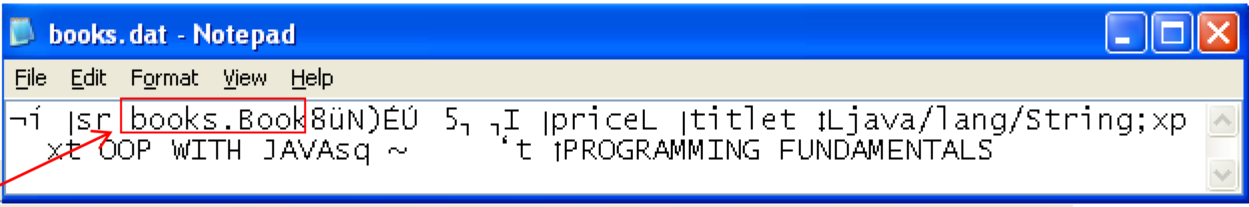
Access Object Files…: Case Study 3– Implementations…



BookList.java

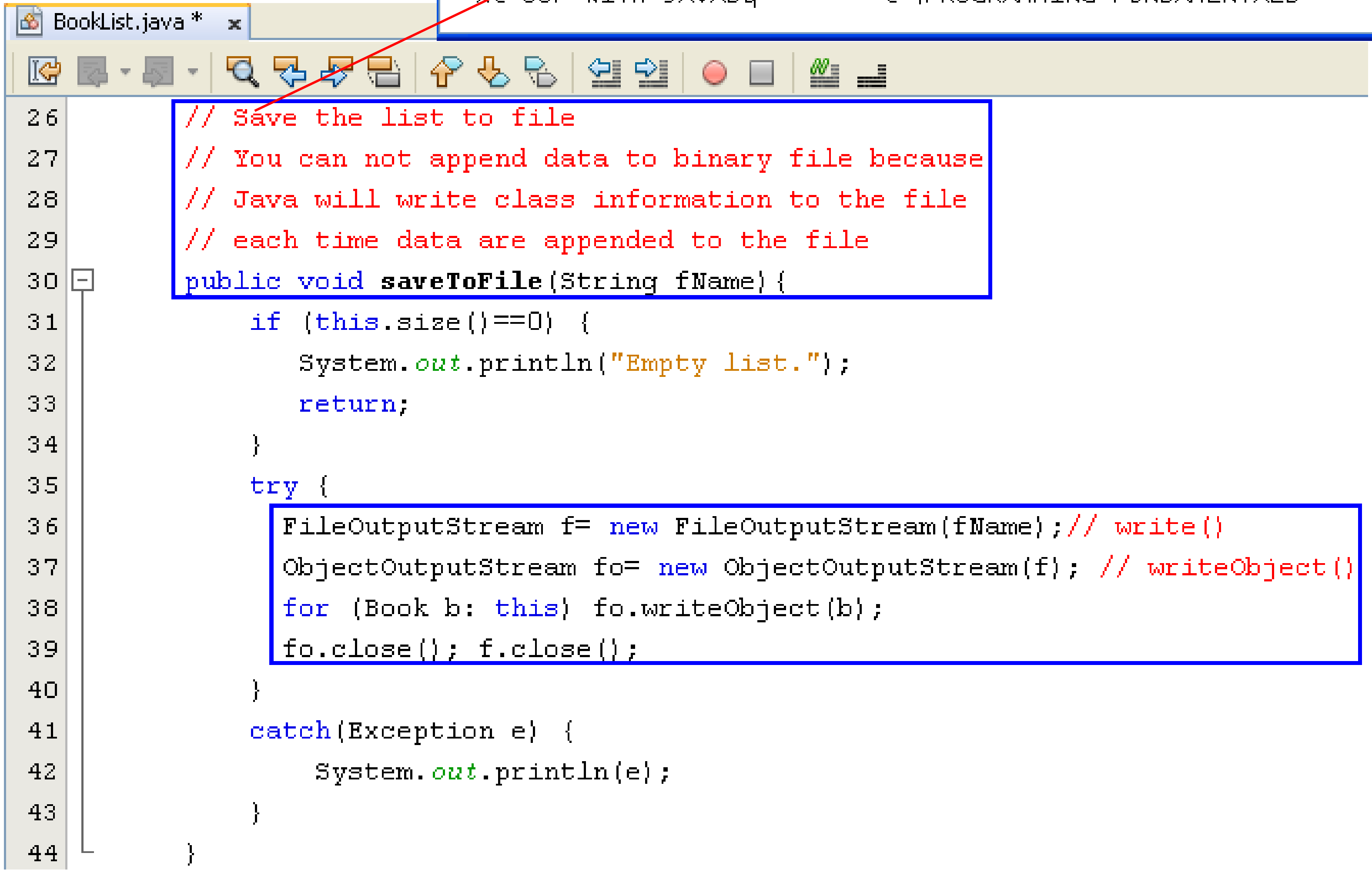
```
/* Class for a book list */
package books;
import java.util.Scanner;
import java.util.Vector;
import java.io.*;
public class BookList extends Vector<Book> {
    Scanner sc= new Scanner (System.in);
  public void loadBookFromFile(String fName){
        // Clear current list before loading codes
        if (this.size()>0)this.clear();
        try {
          File f= new File(fName); // checking the file
          if (!f.exists()) return;
          FileInputStream fi= new FileInputStream(f);// read()
          ObjectInputStream fo= new ObjectInputStream(fi); // readObject()
          Book b;
          while ( (b=(Book)(fo.readObject())) != null) {
              this.add(b);
          }
          fo.close(); fi.close();
        }
        catch(Exception e) {
            System.out.println(e);
        }
  }
```

Access Object Files…: Case Study 3– Implementations…



books.dat - Notepad

```
¬í  ∣sr  books.Book8üN)ÉÚ  5┐ ┐I  ∣priceL  ∣titlet  ↕Ljava/lang/String;xp
  xt  OOP WITH JAVAsq ~       ‛t  ↕PROGRAMMING FUNDAMENTALS
```

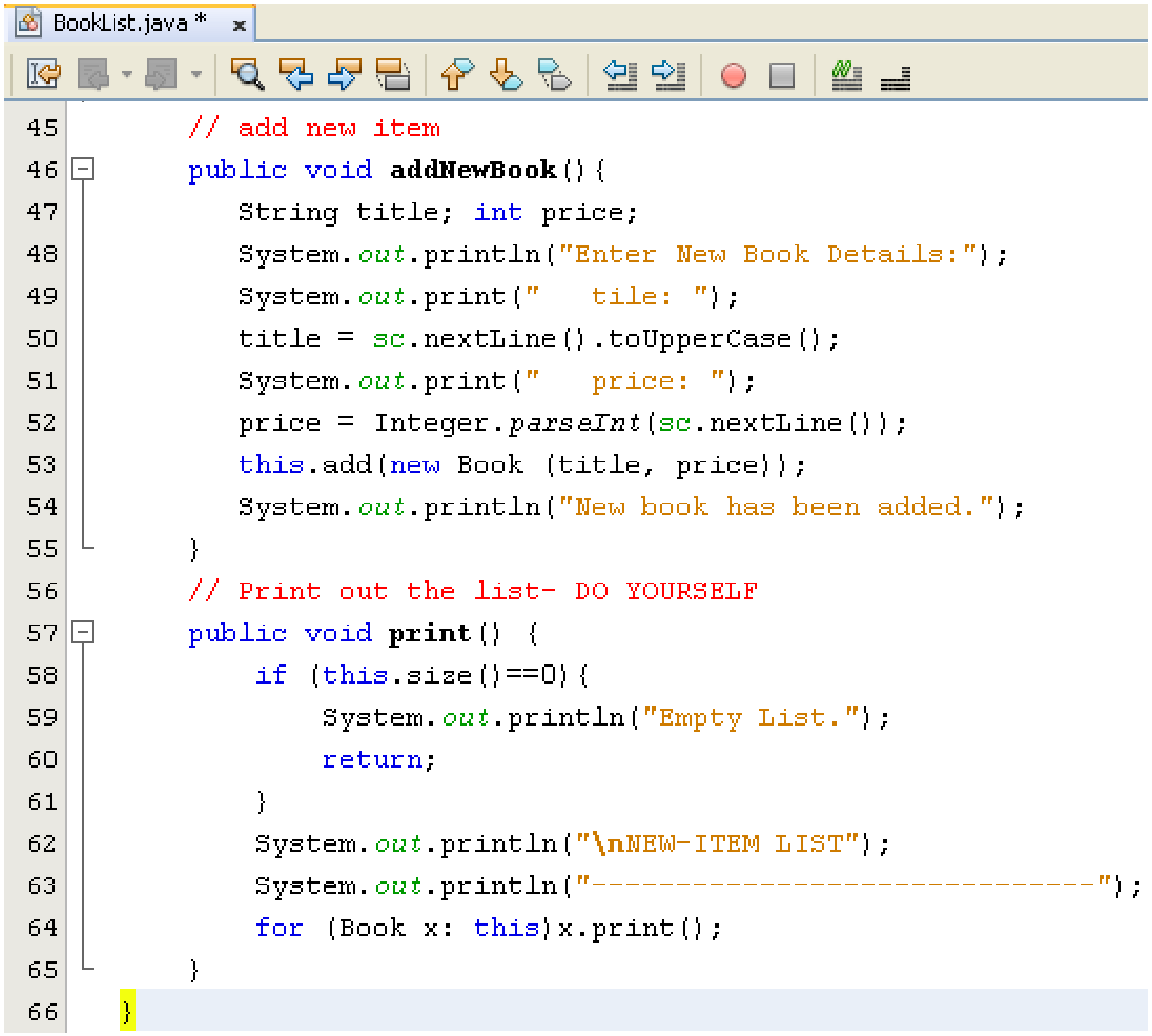


BookList.java *

```
    // Save the list to file
    // You can not append data to binary file because
    // Java will write class information to the file
    // each time data are appended to the file
    public void saveToFile(String fName){
        if (this.size()==0) {
            System.out.println("Empty list.");
            return;
        }
        try {
            FileOutputStream f= new FileOutputStream(fName);// write()
            ObjectOutputStream fo= new ObjectOutputStream(f); // writeObject()
            for (Book b: this) fo.writeObject(b);
            fo.close(); f.close();
        }
        catch(Exception e) {
             System.out.println(e);
        }
    }
```

TRƯỜNG ĐẠI HỌC FPT

Access Object Files…: Case Study 3– Implementations…

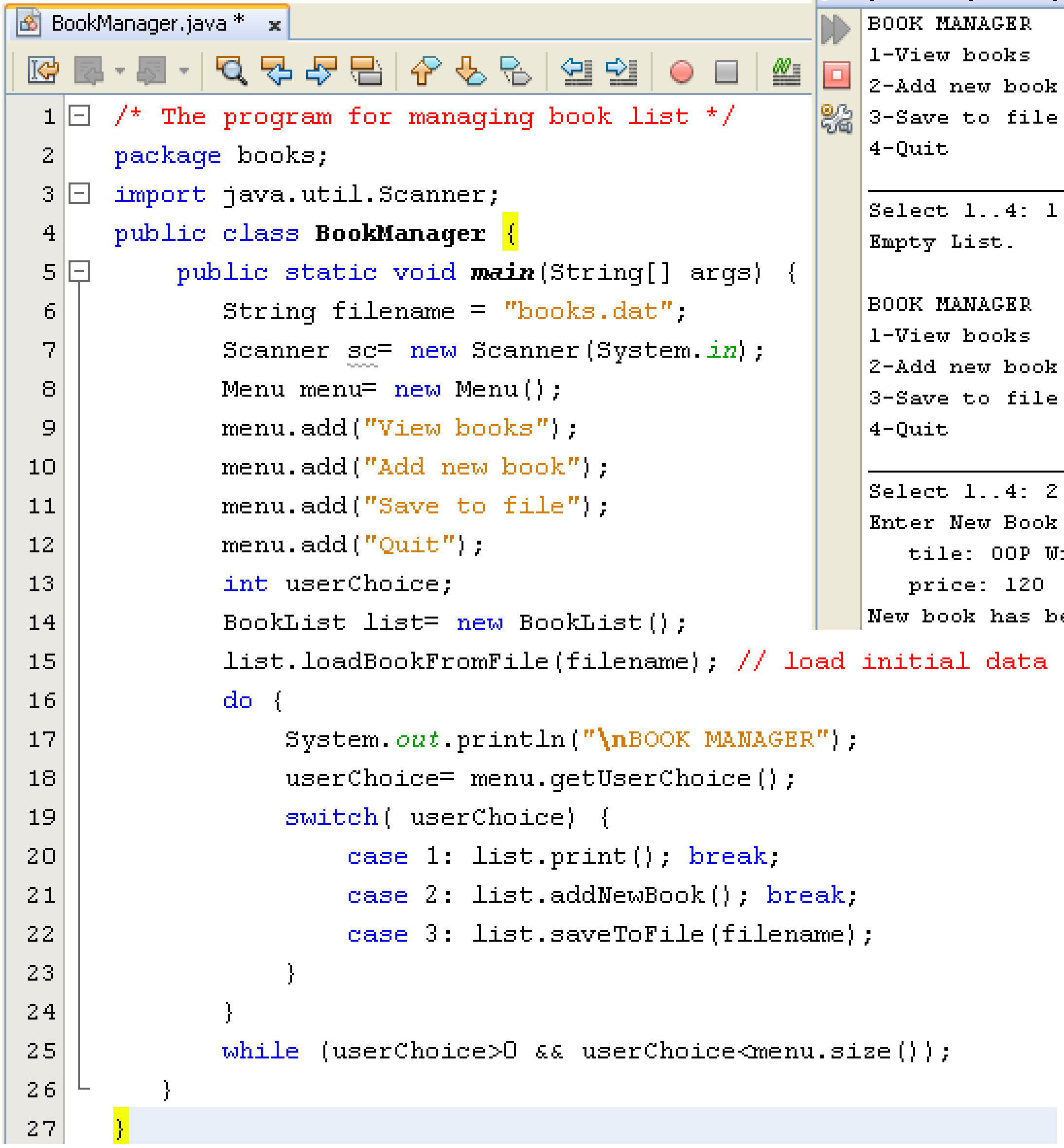


BookList.java *

```
    // add new item
    public void addNewBook(){
        String title; int price;
        System.out.println("Enter New Book Details:");
        System.out.print("    tile: ");
        title = sc.nextLine().toUpperCase();
        System.out.print("    price: ");
        price = Integer.parseInt(sc.nextLine());
        this.add(new Book (title, price));
        System.out.println("New book has been added.");
    }
    // Print out the list- DO YOURSELF
    public void print()  {
         if (this.size()==0){
             System.out.println("Empty List.");
             return;
         }
         System.out.println("\nNEW-ITEM LIST");
         System.out.println("-------------------------------");
         for (Book x: this)x.print();
    }
}
```

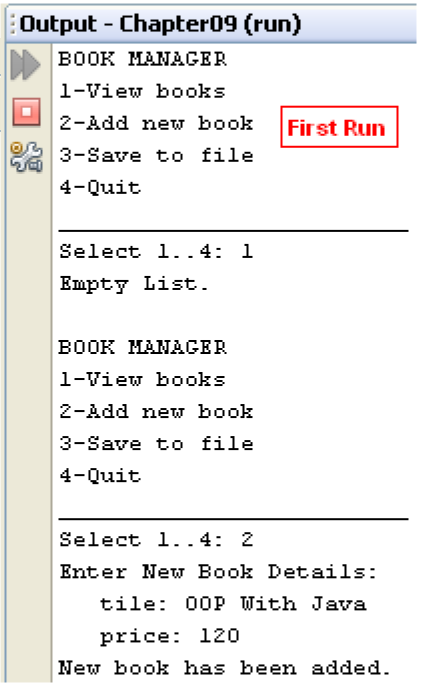
TRƯỜNG ĐẠI HỌC FPT

Access Object Files…: Case Study 5 – Implementations…



BookManager.java *

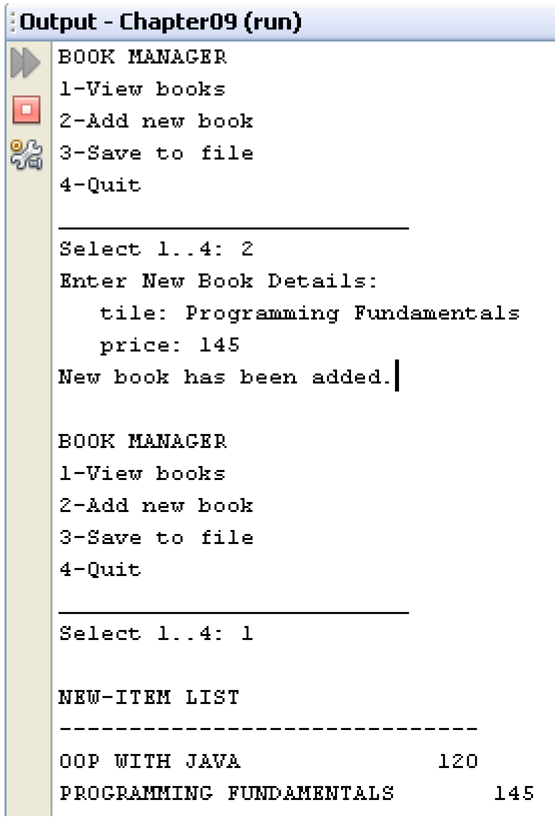
```java
/* The program for managing book list */
package books;
import java.util.Scanner;
public class BookManager {
    public static void main(String[] args) {
        String filename = "books.dat";
        Scanner sc= new Scanner(System.in);
        Menu menu= new Menu();
        menu.add("View books");
        menu.add("Add new book");
        menu.add("Save to file");
        menu.add("Quit");
        int userChoice;
        BookList list= new BookList();
        list.loadBookFromFile(filename); // load initial data
        do {
            System.out.println("\nBOOK MANAGER");
            userChoice= menu.getUserChoice();
            switch( userChoice) {
                case 1: list.print(); break;
                case 2: list.addNewBook(); break;
                case 3: list.saveToFile(filename);
            }
        }
        while (userChoice>0 && userChoice<menu.size());
    }
}
```



Output - Chapter09 (run)

First Run

```
BOOK MANAGER
1-View books
2-Add new book
3-Save to file
4-Quit
_______________________________________
Select 1..4: 1
Empty List.

BOOK MANAGER
1-View books
2-Add new book
3-Save to file
4-Quit
_______________________________________
Select 1..4: 2
Enter New Book Details:
    tile: OOP With Java
    price: 120
New book has been added.
```



Output - Chapter09 (run)

```
BOOK MANAGER
1-View books
2-Add new book
3-Save to file
4-Quit
_______________________________________
Select 1..4: 2
Enter New Book Details:
    tile: Programming Fundamentals
    price: 145
New book has been added.

BOOK MANAGER
1-View books
2-Add new book
3-Save to file
4-Quit
_______________________________________
Select 1..4: 1

NEW-ITEM LIST
--------------------------------
OOP WITH JAVA                120
PROGRAMMING FUNDAMENTALS        145
```

Summary

- Text, UTF, and Unicode
- Accessing metadata of directories/files (java.io.File)
- Text Streams, Reader, and Writer
- The java.io.RandomAccessFile Class
- Binary file Input and Output (low and high-level)
- Object Streams and Serializable